UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

INGENIERÍA DE TELECOMUNICACIONES

PROYECTO FIN DE CARRERA

ESTUDIO Y DESARROLLO DE UN PROTOCOLO DE ENCAMINAMIENTO GEOGRÁFICO Y SU INTEGRACIÓN CON IPV6

Autor: Víctor Sandonís Consuegra
Tutor: María Calderón Pastor
Diciembre 2009

Agradecimientos

Podríamos decir que este proyecto fin de carrera es el punto y final de una etapa de mi vida. El camino seguido para llegar hasta aquí no ha sido fácil, a veces ha sido como pedalear en una cuesta arriba llena de baches. Por eso tengo que dar las gracias a todas esas personas que me han dado un buen empujón en los malos momentos y que se han alegrado conmigo en los buenos, porque no todo ha sido malo, también ha habido buenos momentos.

En primer lugar quiero dar la gracias a mi familia directa. Papa, Mama, Tata y Rocío, sin vosotros no habría sido posible llegar hasta aquí. Vosotros habéis sabido aconsejarme y transmitirme ese espíritu de superación que muchas veces he necesitado. Habéis sido las piezas claves del puzle. Siempre habéis estado ahí cuando lo he necesitado. Vuestro apoyo y cariño ha sido fundamental. Significa mucho todo lo que habéis hecho por mí y no sé cómo expresarlo en palabras. Gracias por vuestra paciencia. Gracias por todo.

También quiero dar las gracias al resto de mi familia por querer lo mejor para mí y apoyarme a lo largo de estos años. Tíos, primos, abuelas, suegra, gracias.

Gracias también a todos mis amigos. Si el que tiene un amigo tiene un tesoro, yo soy más rico que Bill Gates. En especial a Ali, Edu, Imane, Gloria, Mamen y Pablo.

También quiero dar las gracias a los amigos de batalla, los amigos de la carrera. Ellos son los que me han acompañado en el camino a lo largo de estos años. En especial a Carlos y Fernando (sin que ningún otro se enfade), ellos han estado luchando conmigo en primera línea. He aprendido mucho de vosotros, gracias.

Tengo que dar las gracias también a María y Carlos. Gracias por vuestros conocimientos, tiempo, paciencia e interés en que todo saliera bien. Este proyecto no habría sido posible sin vuestra ayuda.

Gracias también a todo aquél que ha continuado leyendo hasta aquí esperando ser encontrado. Son muchas las personas a las que tengo que dar las gracias y seguro que alguien se me ha olvidado. Siento no haberme acordado de ti.

Resumen

El proyecto fin de carrera ha consistido en la implementación de un protocolo de encaminamiento geográfico y su integración con IPv6. Se ha modificado la torre de protocolos de red del sistema operativo Linux para que sea capaz de trabajar en un entorno de comunicaciones intervehiculares. De esta manera, el sistema operativo tiene la capacidad de encaminar paquetes IPv6 utilizando las posiciones geográficas que ocupan los nodos de la red. *Click Modular Router* es la herramienta que se ha empleado para desarrollar la implementación y ejecutar el protocolo de encaminamiento geográfico como un módulo del *kernel* de Linux. Dentro del estudio de la integración de IPv6 con el protocolo de encaminamiento geográfico se ha puesto especial interés en el método por el que los nodos de la red son capaces de configurar de manera automática una dirección IPv6 para sus interfaces. Este método consiste en la adaptación del mecanismo de IPv6 *Stateless Address Autoconfiguration* a entornos de redes vehiculares. Como evaluación de la implementación realizada, se presentan una serie de pruebas y medidas de tiempo que se realizan sobre un software de virtualización. El software de virtualización utilizado es *VMWare Server 2.0*, que permite crear un entorno de pruebas utilizando un único ordenador. Finalmente, se extraen las conclusiones de la realización del proyecto y se plantean posibles ampliaciones del mismo para trabajos futuros.

Abstract

This final degree project involves the implementation of a geographic routing protocol and the integration of the IPv6 stack over it. The network protocol stack of Linux operating system has been modified to be able to work in an inter-vehicle communications environment. Thus, the operating system can route IPv6 packets using the geographical position of the network nodes. Click Modular router is the tool which has been used to develop the implementation and execute the geographic routing protocol as a Linux kernel module. In the study of the integration of IPv6 stack over the geographic routing protocol, we have emphasized the method by which network nodes are able to automatically configure an IPv6 address for its interfaces. This method involves the adaptation of IPv6 Stateless Address Autoconfiguration mechanism for vehicular network environments. As evaluation of the implementation, we present a set of tests and measurements that are performed on virtualization software. VMWare Server 2.0 is the virtualization software which has been used and allows us to create a test environment using a single computer. Finally, we draw conclusions from the project and propose possible extensions of it for future works.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

El proyecto consiste en la implementación de un protocolo de encaminamiento geográfico y su integración con IPv6. La idea es modificar la torre de protocolos de Linux para dotar al sistema operativo de la capacidad de encaminar paquetes IPv6 utilizando las posiciones geográficas que ocupan los nodos de la red.

1.1- Motivación del proyecto

Los protocolos de encaminamiento geográfico se usan en redes ad hoc formadas por vehículos para poder establecer comunicaciones entre ellos. Este tipo de redes reciben el nombre de redes VANET (*Vehicular Ad hoc NETworks*). La investigación en este tipo de redes ha ido creciendo desde la aparición de las tecnologías inalámbricas IEEE 802.11. Este tipo de tecnologías, permiten desarrollar sistemas en los que los vehículos puedan comunicarse entre sí con dispositivos que, sin presentar una complejidad extrema, tienen capacidades suficientes a un bajo coste. De hecho, existe una variación del estándar IEEE 802.11, denominado IEEE 802.11p en el que se introducen algunas mejoras para las comunicaciones *wireless* en un entorno de redes vehiculares.

Los protocolos de encaminamiento geográfico parecen ser la mejor alternativa para encaminar los paquetes en redes VANET. Esto se debe a que al contrario de los protocolos clásicos de encaminamiento que se basan en la topología de la red y en el intercambio de información sobre la existencia de enlaces entre nodos, los protocolos de encaminamiento geográfico únicamente necesitan información sobre la posición geográfica de los nodos de la red para encaminar los paquetes. Por ello, los protocolos de encaminamiento geográfico precisan de una menor señalización y se comportan mejor ante la movilidad de la red. Hay que mencionar que los nodos de la red pueden obtener su posición geográfica gracias a la disponibilidad de sistemas GPS de bajo coste.

Hoy en día, las redes VANET están suscitando un gran interés debido a que son consideradas una de las alternativas más adecuadas para mejorar la seguridad vial. El intercambio de información entre vehículos permite desarrollar aplicaciones con las que se pueden evitar accidentes de tráfico. Informar con antelación al conductor de un vehículo de atascos repentinos o frenazos bruscos, pronosticar colisiones inevitables entre vehículos para tomar las medidas oportunas antes del impacto (mecanismos de frenado automático, activación de los airbags antes del impacto, etc.) o informar de puntos conflictivos y zonas peligrosas en la carretera, son sólo algunos de los ejemplos de aplicaciones que sirven para mejorar la seguridad vial.

Aunque son las más importantes desde el punto de vista social, las aplicaciones de seguridad vial no son las únicas que permiten desarrollar estas redes. Otros tipos de aplicaciones de gestión del tráfico, información o de entretenimiento son posibles. Además, también es posible que los vehículos se puedan conectar a la infraestructura de red de internet mediante puntos de conexión situados a lo largo de las carreteras.

El hecho de que los vehículos puedan establecer comunicaciones con cualquier ordenador de internet implica que éstos deben ser capaces de utilizar los protocolos que se utilizan en internet. Por ello, es necesaria la integración de estos protocolos con el de encaminamiento geográfico. Debido a que en los últimos años se está promoviendo el desarrollo de IPv6 y a que los organismos más importantes que tratan con temas de redes vehiculares han incluido IPv6 en sus pilas de protocolos, en el proyecto se estudia la integración de IPv6 sobre el nivel de encaminamiento geográfico.

En particular, dentro de la integración de IPv6 sobre el protocolo de encaminamiento geográfico, se presentan las capacidades que deben tener los nodos de la red vehicular para poder configurar de manera automática una dirección IPv6. Hay que tener en cuenta que, uno de los puntos clave en la interconexión de la red vehicular con internet, es la manera en que sus nodos configuran las direcciones IPv6 de sus interfaces de red.

1.2- Objetivos del proyecto

El proyecto tiene dos grandes objetivos:

- La implementación de un protocolo de encaminamiento geográfico que permita el encaminamiento de paquetes basándose en la posición geográfica de los nodos que forman la red.

- La integración de IPv6 sobre el protocolo de encaminamiento geográfico y el estudio del método con el que los nodos de la red son capaces de configurar una dirección IPv6 global de manera automática.

Con la implementación del protocolo de encaminamiento geográfico se pretende:

- Poder realizar encaminamiento de paquetes dirigidos a un único nodo que se encuentra en una determinada posición geográfica. El encaminamiento del paquete se debe realizar en función de la posición geográfica de los nodos de la red. Es lo que se denomina entrega *geounicast*.

- Poder realizar encaminamiento de paquetes que van dirigidos a un grupo de nodos que se encuentran en el interior de una determinada región geográfica. El encaminamiento del paquete se realiza en función de la posición geográfica de los nodos de la red. Es lo que se denomina entrega *geobroadcast*.

La integración de IPv6 sobre el protocolo de encaminamiento geográfico debe seguir estas premisas:

- Se debe permitir a los nodos de la red vehicular soportar IPv6 y así, poder comunicarse con otros nodos de la red vehicular o *host* de internet utilizando IPv6.

- Los vehículos de la red deben ser capaces de configurar la dirección IPv6 de sus interfaces de manera automática.

- Los cambios necesarios sobre el nivel IPv6 de la pila de protocolos de Linux de los nodos de la red vehicular deben ser los mínimos posibles.

- La comunicación entre un nodo de la red vehicular y un *host* de internet debe poderse mantener sin que el último modifique el funcionamiento de su nivel IPv6.

1.3- Contenido de la memoria

La memoria se estructura en siete capítulos y seis apéndices.

El capítulo1 sirve de introducción al proyecto. Se divide en tres apartados donde se detallan las motivaciones que justifican la realización del proyecto, los objetivos planteados que se pretenden cumplir y, el apartado actual, que presenta cómo se estructura la memoria.

El capítulo 2 es el estado del arte. El capítulo se divide en cinco apartados que sirven de introducción teórica a la base de conocimientos sobre los que se apoya el proyecto.

El capítulo 3, "Diseño del protocolo de encaminamiento geográfico y de la integración con IPv6" se divide en tres apartados. Los dos primeros explican el diseño del protocolo de encaminamiento geográfico y el último, describe cómo conseguir que el nivel IPv6 pueda funcionar sobre el nivel de encaminamiento geográfico.

En el capítulo 4 titulado "Herramienta utilizada en el desarrollo del proyecto: *Click Modular Router*", se presenta la herramienta básica que se ha utilizado para realizar la implementación del protocolo de encaminamiento geográfico y lograr su integración con IPv6. El capítulo se divide en cinco apartados donde se describen los aspectos más importantes para entender el funcionamiento básico de la herramienta.

El capítulo 5 titulado "Implementación del nivel de encaminamiento geográfico y su integración con IPv6", se divide en cuatro apartados. En ellos se muestra cómo se ha dividido la implementación en módulos y se explica en detalle cada uno de ellos.

El capítulo 6, "Evaluación de la implementación", muestra las pruebas realizadas sobre la implementación que permiten obtener las conclusiones de la elaboración del proyecto.

El capítulo 7, "Conclusiones y trabajos futuros", repasa los objetivos del proyecto y analiza su cumplimiento. Es una valoración de los resultados obtenidos. Por otro lado, también se deja una puerta abierta a posibles ampliaciones futuras del proyecto.

En los apéndices, se muestra la división en tareas y planificación temporal seguida en la realización del proyecto fin de carrera, el manual de instalación de la implementación, un pequeño manual de ejecución, el archivo de configuración y el esquema de elementos de la implementación, una descripción del contenido del CDROM con el código de la implementación y la bibliografía del proyecto.

Capítulo 2

Estado del Arte

2.1- Definición de redes VANET y comunicaciones C2C

Las redes VANET (*Vehicular Ad hoc NETworks*) son un caso particular de redes *ad hoc*. Una red *ad hoc* está compuesta por un conjunto de nodos capaces de comunicarse a través de una interfaz inalámbrica de manera descentralizada, es decir, cada nodo está preparado para reenviar datos a los demás y la decisión sobre qué nodos reenvían los datos se toma de forma dinámica en función de la conectividad de la red. De esta manera, un nodo puede comunicarse con otro que se encuentre fuera de su rango de cobertura por medio de una comunicación multisalto, es decir, se establece un puente entre origen y destino a través de nodos intermediarios y la información se reenvía de nodo en nodo desde el emisor hasta el receptor.

Lo que diferencia a las redes VANET de otro tipo de redes *ad hoc* es que sus nodos se mueven a mayor velocidad y siguiendo un patrón restringido, es decir, los nodos o vehículos se mueven sobre carreteras con una determinada topología y siguiendo una serie de normas. Además, en las redes VANET, los vehículos o nodos cuentan con grandes capacidades de memoria y procesamiento, al contrario, por ejemplo, que en las redes MANET (*Mobile Ad hoc NETworks*) donde los nodos tienen un patrón de movilidad arbitrario y la capacidad de memoria y computación están limitadas.

Por lo tanto, las comunicaciones C2C (*Car 2 Car*) o intervehiculares son aquellas que, utilizando las redes VANET, permiten el intercambio de información entre vehículos.

2.2- Motivación de las comunicaciones C2C

Las comunicaciones entre vehículos abren un gran abanico de posibles aplicaciones para mejorar la seguridad al volante, aumentar la eficiencia del tráfico y proporcionar información útil a los conductores.

De entre las principales motivaciones de las comunicaciones C2C podemos destacar:

- Mejora de la seguridad vial: las comunicaciones entre vehículos pueden ayudar a mejorar la seguridad de los conductores. La idea es que los vehículos compartan entre sí información sobre su posición, velocidad y dirección, de manera que ésta se pueda utilizar para evitar colisiones por frenazos bruscos o atascos repentinos, informando con antelación al conductor. Continuando con esta idea, cuando el sistema determine una colisión inevitable entre vehículos, se tomarán las medidas oportunas con mecanismos

de frenado automático o activación de los airbags antes del impacto. También se pueden utilizar las comunicaciones C2C para informar al resto de vehículos de puntos conflictivos o zonas peligrosas en la carretera (por ejemplo hielo en un tramo de vía) de forma que los conductores puedan ser informados con suficiente antelación. Estos son sólo algunos ejemplos de las numerosas aplicaciones de las comunicaciones C2C en el ámbito de la seguridad vial.

- Mejora de la eficiencia de tráfico: Como los vehículos comunican su posición geográfica, se puede utilizar esta información en sistemas que calculen las condiciones de tráfico y proporcionen las mejores rutas a seguir por los conductores en tiempo real. Otro caso de aplicación de las comunicaciones C2C es aquel en el que se facilita a los vehículos la incorporación a una autovía o autopista. Para ello, teniendo en cuenta las condiciones de circulación de la vía, se informa a los conductores para que realicen una incorporación adecuada sin interrumpir la fluidez del tráfico. Otra posible aplicación dentro de este ámbito es la que se podría denominar *anuncio de velocidad óptima*. *Anuncio de velocidad óptima* consiste en que cuando un vehículo se aproxima a una intersección regulada por semáforos, éste recibe información del estado y temporización de los mismos. Así, se puede indicar al conductor la velocidad adecuada que debe llevar para evitar parar innecesariamente en la intersección.

- Provisión de servicios de información: se puede proporcionar acceso a internet a los vehículos de forma que los conductores puedan acceder a cualquier información de su interés o utilizar cualquiera de los servicios comunes de las redes IP. Las comunicaciones C2C también permiten ofrecer un servicio de notificaciones de puntos de interés. Por ejemplo, que una gasolinera difunda información sobre sus precios a aquellos vehículos que circulen en su cercanía o un servicio por el cual un conductor pueda saber la ubicación de los parkings más cercanos a su posición y su estado de ocupación, precios, etc.

La motivación más importante de las citadas anteriormente es la de mejorar la seguridad vial. Es posible que con las comunicaciones intervehiculares se consigan reducir de forma muy significativa el número de accidentes de tráfico y número de víctimas, estadísticas que preocupan cada vez más a la sociedad y a los gobiernos, y donde las comunicaciones C2C pueden ser una alternativa a tener muy en cuenta.

2.3- Prerrequisitos y restricciones de las comunicaciones C2C

Las comunicaciones C2C abren un mercado con grandes expectativas de futuro donde, sobre todo las aplicaciones sobre seguridad vial y control de tráfico, tienen un gran potencial. Pero el desarrollo de un sistema de comunicaciones intervehiculares no está exento de problemas y cuenta con una serie de prerrequisitos y restricciones, que varía en función de la aplicación concreta y que se pueden separar en restricciones económicas y técnicas.

2.3.1- Prerrequisitos y restricciones económicas

Tal y como se ha planteado el sistema, las comunicaciones intervehiculares precisan de una cierta penetración en el mercado antes de que puedan funcionar correctamente. Se estima que la penetración en el mercado debe ser al menos del 10% para aplicaciones de seguridad vial y de al menos el 5% para aplicaciones de difusión de información de tráfico. Incluso en el mejor de los casos en el que todos los coches que salen de las fábricas estuvieran equipados con esta tecnología, se tardaría año y medio en llegar a una penetración del 10% y más de 6 años en llegar al 50% [3]. Éste es uno de los grandes problemas de las comunicaciones entre vehículos. Al tratarse de un sistema cooperativo, el consumidor no percibe de inmediato los beneficios del sistema, sino que, hasta que una cantidad de vehículos suficiente no tengan instalados los equipos, el sistema no puede funcionar correctamente. Este motivo frena que los clientes equipen a sus coches con estas tecnologías.

Por este motivo, es necesaria una iniciativa conjunta de los fabricantes de coches, organizaciones científicas, gobiernos y organizaciones de estandarización para que la introducción en el mercado de esta tecnología tenga éxito.

2.3.2- Prerrequisitos y restricciones técnicas

Existen también una serie de problemas técnicos que es necesario solventar como son:

- Privacidad: debido a que se envía y comparte información entre diferentes nodos es necesario garantizar la privacidad de los usuarios del sistema de manera que, por ejemplo, no se pueda rastrear la posición de un conductor. Para ello se plantea la utilización de una serie de identificadores temporales.

- Fiabilidad de los datos: el sistema debe asegurar que los datos sean fiables, y que nadie pueda enviar información malintencionadamente. Para garantizar que los usuarios puedan confiar en los datos que reciben se propone la utilización de firmas digitales y certificados.

- Banda de frecuencias protegida: para aplicaciones de seguridad vial, las comunicaciones deben ser robustas, garantizar una mínima latencia y cierta fiabilidad. Es necesario que un mensaje de seguridad crítico llegue a todos los receptores a tiempo mientras que se juega *on-line* en el vehículo o se descarga un archivo de internet, etc. Esto no se puede conseguir si aquellas aplicaciones que tratan con temas de seguridad vial utilizan la misma banda de frecuencias que las aplicaciones que no están destinadas a la seguridad vial, ya que éstas últimas, podrían retrasar la entrega de información crítica. Por ello, existe una banda de frecuencias reservada para aplicaciones de seguridad críticas entre 5,885 GHz y 5,905 GHz. La asignación de bandas para el resto de aplicaciones es un trabajo que todavía no está concluido.

- Escalabilidad: es necesario que el sistema funcione correctamente con diferentes densidades de vehículos. Debe ser capaz de trabajar en el caso en el que la densidad de

vehículos es baja y los rangos de cobertura de las antenas de los vehículos dificulte la comunicación entre ellos, como por ejemplo, en un entorno rural. Del mismo modo, debe funcionar correctamente en el caso en el que el tráfico es muy intenso, como puede ser en un atasco en una ciudad, donde la multitud de mensajes enviados puede comprometer el ancho de banda disponible. Es necesario tratar con los diferentes requisitos tecnológicos que presentan ambos casos.

- Equipamiento obligatorio: para que todas las aplicaciones propuestas funcionen, es necesario que todos los coches dispongan de una serie de dispositivos obligatorios que proporcionen por ejemplo: datos de posición, velocidad, dirección, etc.

2.4- Arquitectura del sistema C2C –CC (*Car 2 Car Communication Consortium*)

El *Car 2 Car Communication Consortium* (C2C-CC) [4] es una asociación formada por los principales fabricantes de automóviles, empresas de dispositivos electrónicos y centros de investigación que tienen como objetivo la estandarización de las interfaces y protocolos de un sistema de comunicaciones entre vehículos.

A continuación se describe la arquitectura del sistema de comunicaciones intervehiculares del *Car 2 Car Communication Consortium* [3].

2.4.1- Elementos del sistema

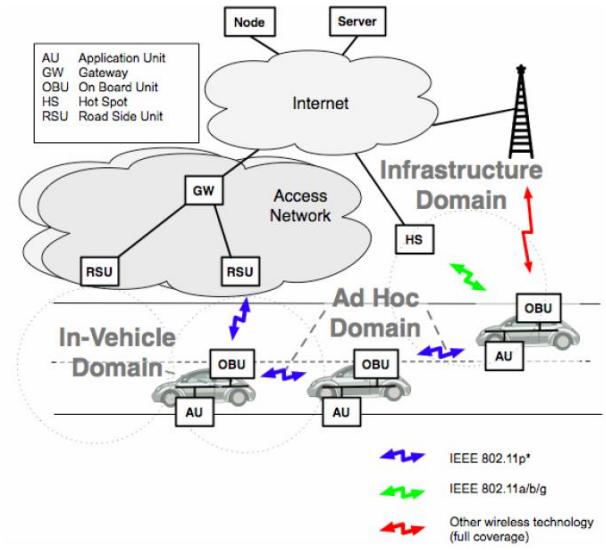

Figura 1 : Arquitectura del sistema C2C-CC

En la figura 1 se pueden ver los elementos que forman el sistema de comunicaciones C2C-CC. El sistema se puede dividir en tres dominios:

Dominio del vehículo:

Este dominio está formado por las unidades de a bordo (OBU, *On-Board Unit*) y las unidades de aplicación (AU, *Aplication Unit*). Las OBUs pueden verse como *routers* móviles con al menos 2 interfaces de red, una para la red interna del vehículo a la que se conectan las unidades de aplicación (AU), y otra, para comunicarse con la red VANET. Una AU es un dispositivo que ejecuta una serie de aplicaciones y que se conecta a la OBU para utilizar sus capacidades de comunicación. Las AUs se pueden conectar con la OBU por medio de un cable o cualquier tecnología *wireless* (*Bluetooth, WUSB, UWB*…).

Dominio *Ad hoc*:

Es lo que se llama red VANET (*Vehicular Ad hoc NETwork*) y está formado por las OBUs de los vehículos y las estaciones fijas situadas al borde de la carretera, las *road-side units* (RSUs). Una OBU está dotada de al menos un sistema de comunicación *wireless* para aplicaciones de seguridad vial (concretamente IEEE 802.11p) y alternativamente por otro tipo de tecnologías de comunicaciones como, por ejemplo, UMTS. Las OBUs forman entre sí una red *ad hoc* móvil (MANET) que permite la comunicación entre nodos de forma distribuida (sin un nodo central que coordine las comunicaciones). Las OBUs se comunican de la siguiente manera: si dos OBUs se encuentran dentro de su rango de cobertura, se pueden comunicar directamente. Si no, los protocolos de encaminamiento geográfico permiten una comunicación multisalto, de manera que los paquetes son retransmitidos por las OBUs hasta que llegan al nodo receptor. Una breve descripción de los protocolos que permiten las comunicaciones cuando no hay "visión directa" entre dos nodos se verá en el apartado 2.5 "Protocolos de encaminamiento geográfico".

Por otro lado, las RSUs son nodos fijos de la red *ad hoc* que tienen la función básica de enviar, recibir y retransmitir paquetes para aumentar el rango de cobertura de la red, algo muy importante en las aplicaciones de seguridad vial. También pueden ofrecer acceso a internet al encontrarse conectadas a la red fija de algún operador. De esta manera, las RSUs permiten a los vehículos estar conectados a la infraestructura fija, de modo que cualquier AU puede conectarse con cualquier *host* de internet.

Dominio de la infraestructura:

Este dominio está formado por las dos unidades básicas de acceso a la red de infraestructura, la RSU y los *Hot Spot* (HS). Los HS son similares a las RSU, pero no participan en ninguna aplicación de seguridad vial. Su uso es comercial y ofrecen servicio de conexión a internet gestionado por algún proveedor privado. Las RSUs, al participar en aplicaciones de seguridad vial, deben estar supervisadas por un organismo público.

Dentro de este dominio nos encontramos también con una autoridad de certificación (CA) que se encarga de emitir certificados digitales a las OBUs y RSUs.

Como último apunte, señalar que cuando una OBU quiere conectarse a internet para aplicaciones que no son de seguridad vial puede hacerlo a través de otras tecnologías (UMTS, GPRS, WiMax, etc.) sin utilizar el acceso por medio de las RSUs o HS.

2.4.2- Principios básicos de las comunicaciones C2C

Las comunicaciones C2C:

- Permiten la difusión de información de manera controlada en espacio y tiempo entre vehículos, algo fundamental en las aplicaciones de seguridad vial.

- Ofrecen servicios de difusión de paquetes similares a los de las redes convencionales de conmutación de paquetes. Se cuenta con envío de paquetes *unicast, multicast, anycast* y *broadcast*, pero adaptados a redes móviles inalámbricas.

Hay dos formas de difundir información en las comunicaciones intervehiculares:

- Difusión centrada en el receptor: cuando un vehículo detecta algún peligro por medio de sus sensores, distribuye dicha información a sus nodos vecinos. Entonces los vecinos combinan la información recibida con la suya propia y redistribuyen la combinación, de nuevo, a sus vecinos. La distribución espacial y temporal es controlada por el nodo receptor porque, es éste, el que decide si enviar o no la información a sus vecinos en función de la importancia de la misma.

- Difusión centrada en el emisor: el nodo emisor es el que decide sobre la difusión de la información estableciendo un área geográfica dentro de la cual la información debe ser entregada. En recepción, los nodos vecinos comprueban si están situados dentro de dicho área geográfica y, en consecuencia, si aceptar y retransmitir el paquete.

Todo esto se consigue, como habíamos mencionado anteriormente, gracias a la utilización de los protocolos de encaminamiento geográfico.

2.4.3- Torre de protocolos

En la figura 2 se muestra la torre de protocolos de una OBU:

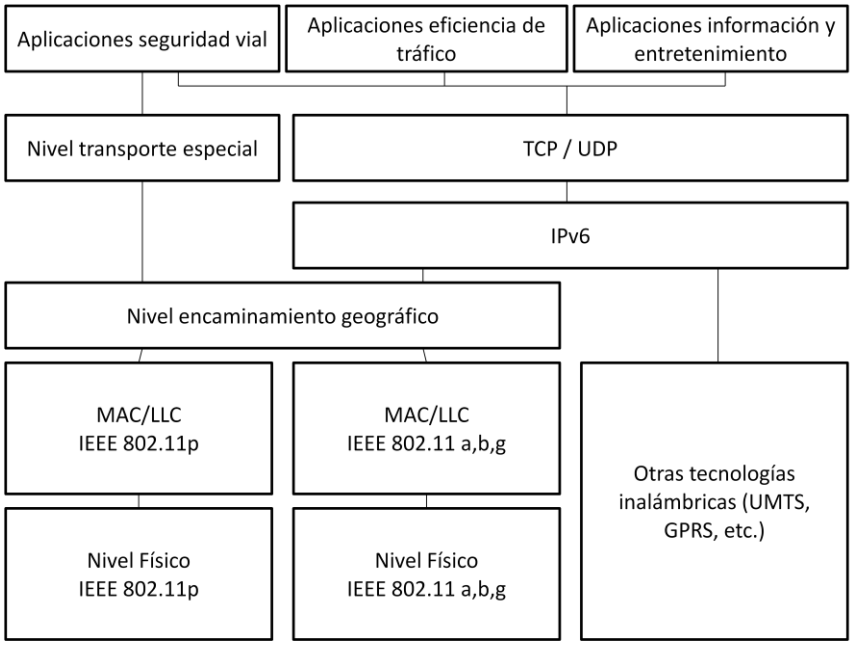

Figura 2 : Arquitectura de protocolos de una OBU

En primer lugar, se distinguen tres tipos de tecnologías radio *wireless*: IEEE 802.11p, IEEE 802.11 a/b/g, que son tecnologías convencionales en redes LAN, y por último otro tipo de tecnologías radio como GPRS, UMTS, HSDPA, etc. Por encima de las capas física y de enlace, tenemos la capa de encaminamiento geográfico que proporciona comunicaciones multisalto basándose en la posición de los nodos de la red. Por otro lado, como se puede apreciar, las aplicaciones que no son críticas usan la pila de protocolos tradicional con TCP y UDP sobre IPv6. Las aplicaciones que requieren de mayor fiabilidad, aquellas ligadas a la seguridad vial,

utilizan los protocolos de transporte especial y de encaminamiento geográfico diseñados expresamente para ello, es decir, están adaptados para su uso en las comunicaciones C2C.

De manera resumida vamos a mostrar las características principales de las capas de nivel de transporte especial y de nivel de encaminamiento geográfico:

Nivel de transporte especial

El protocolo de nivel de transporte especial provee de diferentes servicios para las comunicaciones C2C como son multiplexación y demultiplexación, comunicación *unicast* orientada a conexión y transferencia de datos de forma fiable de acuerdo a las necesidades de las aplicaciones dedicadas a la seguridad vial. Cabe destacar una función adicional de esta capa, que es la de combinar los datos provenientes de distintas aplicaciones y agruparlos de forma que puedan ser enviados en un único paquete.

Nivel de encaminamiento geográfico

Esta capa se encarga de difundir y encaminar los paquetes de información según los principios del encaminamiento geográfico y sus protocolos. En el apartado 2.5 “Protocolos de encaminamiento geográfico” se ilustran los mecanismos de este nivel y los servicios que provee a niveles superiores.

2.5- Protocolos de encaminamiento geográfico

Geocasting se define como la entrega de un paquete a los nodos que se encuentran dentro de una determinada región geográfica. De esta manera, usando información de la posición geográfica en la que se encuentran los distintos nodos que forman la red, se puede enviar un paquete a aquellos nodos que se encuentren dentro de un área geográfica objetivo. Sin embargo, esto no quiere decir que el mensaje tenga que ser entregado a todos los nodos que se encuentran dentro de dicha región geográfica (*Geo-Broadcast*), sino que el paquete puede ir dirigido a un nodo arbitrario de este grupo (*Geo-Anycast*), o a un solo de destinatario concreto (*Geo-Unicast*). Un caso sencillo para entender lo anterior en el ámbito de las redes vehiculares es la situación en la que, por ejemplo, se desea informar de un accidente de tráfico en una autovía a los vehículos que circulan por ella, pero con la restricción de que el mensaje debe ser entregado únicamente a los conductores que circulan en el sentido en el que se ha producido el accidente. Por otro lado, hay que tener en cuenta que cada nodo puede obtener su situación espacial en cada momento gracias a los sistemas de localización GPS actuales o utilizando cualquier otra técnica de localización.

Dentro de los protocolos de encaminamiento en redes *ad hoc*, en función de la información utilizada para encaminar los paquetes, podemos distinguir entre aquellos que están basados en la topología de la red (*topology-based*), y aquellos que se basan en la posición geográfica de los nodos de la red (*position-based*).

Los protocolos clásicos para redes *ad hoc* están basados en la topología de la red, por lo que las decisiones de encaminamiento se basan en los distintos enlaces disponibles entre los nodos de la red. En este tipo de protocolos, los nodos construyen unas tablas con las rutas hacia cada posible nodo destino. En un escenario de redes móviles en el que la topología de la red está en constante

cambio, no es una solución eficiente ya que los nodos intercambiarían constantemente paquetes de control para mantener la tabla de rutas actualizada y se generaría una gran sobrecarga en la red.

En los protocolos basados en posición, los nodos guardan información sobre la posición geográfica de los nodos vecinos y la utilizan para encaminar los paquetes. Esta solución es más eficiente que la anterior ya que los paquetes que intercambian los nodos para saber la posición de los vecinos provoca menor sobrecarga de tráfico, puesto que la información necesaria para construir las tablas de localización es menor. Sin embargo, de esta manera, es necesario que los nodos de la red estén provistos de un sistema GPS para conocer su posición geográfica.

Por este motivo, en este proyecto, nos vamos a centrar en los protocolos de encaminamiento geográfico. Cabe aclarar que dentro de este grupo de protocolos, la funcionalidad proporcionada por cada uno de ellos no es exactamente la misma. En algunos casos únicamente proporcionan servicios *geobroadcast* que hacen que no sea necesario disponer de tabla de localización mientras que en otros casos además de servicios *geobroadcast* hacen posible envío de datos a un único nodo de la VANET, *geounicast*. A continuación se detallan las características de los protocolos de encaminamiento geográfico basados en posición más importantes.

2.5.1- Inundación simple (*Simple flooding*)

Inundación simple [5] se utiliza para distribuir envíos *geobroadcast* dentro de una región geográfica determinada. Su funcionamiento puede resumirse en que se inunda toda la red independientemente de donde vaya dirigido el paquete. Cuando un nodo recibe un paquete, éste comprueba si se encuentra dentro de la región destino (que se encuentra incluida en el paquete). En caso afirmativo, además de ser uno de los destinatarios, retransmite el paquete a todos los nodos vecinos que se encuentran dentro de su rango de alcance. Si el nodo se encuentra fuera de la región destino o el paquete ya había sido recibido anteriormente, éste se descarta para evitar bucles y para que la inundación cese. Esta técnica es robusta, pero poco eficiente debido al gran número de retransmisiones redundantes innecesarias para el envío de un paquete, incrementando así, el tráfico ofrecido en la red. Además de provocar mayor congestión, se utiliza ancho de banda y potencia en los transmisores de forma innecesaria.

2.5.2- DREAM

Distance Routing Effect Algorithm for Mobility, DREAM [7], es un protocolo de encaminamiento que está basado en la posición geográfica de los nodos de la red. DREAM utiliza dos algoritmos, uno para difundir los paquetes de control con la información de localización geográfica y otro para encaminar los paquetes de datos.

La difusión de los paquetes de control está basada en inundación restringida. Todos los nodos mantienen tablas que contienen la localización geográfica de todos los vecinos. De esta manera, cada nodo, periódicamente, difunde su posición geográfica a los vecinos para que éstos actualicen sus tablas. La inundación se ve restringida porque los nodos establecen un límite de

distancia de cobertura y el paquete de control, como máximo, podrá alcanzar este límite. La inundación también se ve restringida en el sentido de la frecuencia con que los nodos informan de su posición geográfica. El periodo con el que los nodos mandan los paquetes de control viene determinado por la tasa de movilidad de la red. Además, con este periodo se puede controlar la sobrecarga producida en la red por los paquetes de control.

El encaminamiento de los paquetes de datos sirve para envíos de tipo *geounicast*. Cuando un nodo S desea transmitir un paquete a un destino D, éste, observando su tabla de localización, selecciona los nodos vecinos que se encuentran en dirección a D y les envía el paquete. Éstos a su vez, realizan la misma operación hasta que el paquete llega al destino saltando de nodo en nodo. En cada salto, los nodos únicamente envían el paquete a los vecinos que se encuentran en el sector del rango de cobertura que está en la dirección del destino, y no a todos los vecinos dentro del radio de alcance del nodo.

2.5.3- LBM (*Location Based Multicast*)

LBM [8] se basa en la inundación de la red pero con la peculiaridad de que define una zona de reenvío (*forwarding zone*) donde los paquetes son reenviados únicamente si el nodo que lo recibe pertenece a dicha zona. Si un nodo recibe el paquete, pero no pertenece a la zona de reenvío, se descarta el paquete para evitar que se inunde completamente la red.

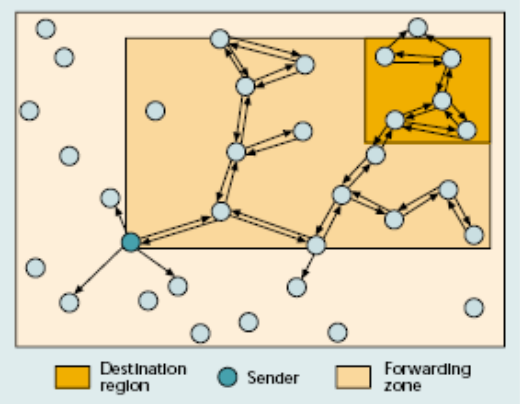

Figura 3 : Ejemplo *LBM-box*

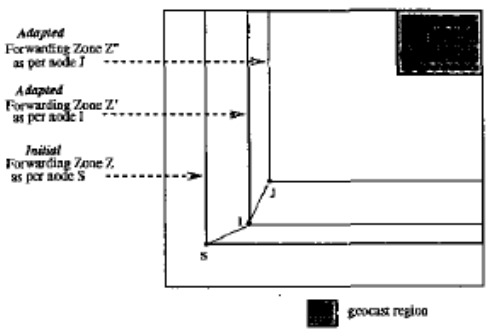

Figura 4 : Ejemplo de zona adaptativa

La zona de reenvío incluye, por lo menos, la región destino del paquete (por lo tanto, se trata de envío *geobroadcast*) y un camino entre el emisor del paquete y la región de destino a la que va dirigida el paquete emitido. De esta manera, cuanto más grande sea esta zona de reenvío, más probabilidades hay de que el mensaje llegue al destino, pero el precio a pagar es un incremento de sobrecarga de tráfico ofrecido en la red.

Dentro del protocolo LBM hay dos opciones que se diferencian en la manera de elegir la forma de la zona de reenvío:

LBM-box: Se define la zona de reenvío como el rectángulo más pequeño que incluye al nodo emisor del paquete y la región destino. Opcionalmente, para aumentar la probabilidad de éxito de recepción de un paquete, se puede incrementar el tamaño de la zona de reenvío aumentando los lados del rectángulo en un factor δ. La zona de reenvío se codifica en cada paquete de modo que un nodo pueda determinar si está situado dentro de la zona de reenvío o no.

En la figura 3, obtenida de [5], se puede apreciar

la manera en la que se define la zona de reenvío en *LBM-box*.

Una forma de mejorar la estrategia *LMB-box* es utilizar una zona adaptativa en la que la zona de reenvío no incluye al emisor inicial del paquete, sino al emisor del último salto del paquete [9]. De esta manera, la zona de reenvío se va adaptando y reduciendo en cada salto que da el paquete de nodo a nodo. Como ejemplo, el que se muestra en la figura 4: el nodo I recibe un paquete emitido por S y lo reenvía a sus nodos vecinos (nodo J), ya que I se encuentra dentro de la zona de reenvío. Cuando el nodo I reenvía el paquete, envía a sus vecinos su localización geográfica y la nueva zona de reenvío Z'. De esta manera, de nuevo, cuando J recibe el paquete procedente de I, éste lo reenvía porque pertenece a la zona de reenvío Z' y vuelve a definir la nueva zona de reenvío. De esta forma, la zona de reenvío va adaptándose según el paquete se acerca a la región destino.

LBM-step: Define la zona de reenvío en función de las distancias de los nodos de la red al centro de la región destino. Cuando un nodo recibe un paquete determina si pertenece a la zona de reenvío calculando la distancia a la que se encuentra del centro geográfico de la región destino. El nodo pertenece a la zona de reenvío si esta distancia (reducida opcionalmente en un factor δ para aumentar la probabilidad de recepción del mensaje) es menor que la almacenada en el mensaje, que no es otra que la del nodo que ha reenviado el paquete en el salto anterior.

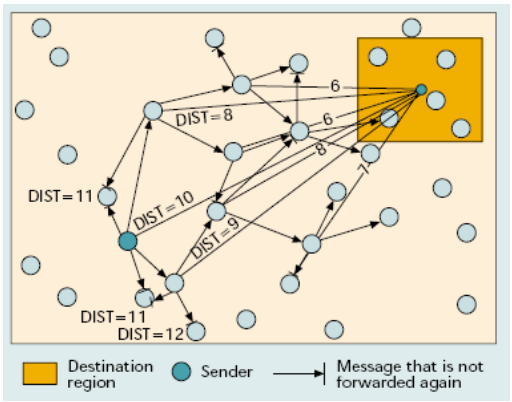

Figura 5 : Ejemplo *LBM-step*

Entonces, si el nodo pertenece a la zona de reenvío, manda a sus nodos vecinos el paquete, introduciendo en él, la nueva distancia al centro de la región destino. Es decir, un nodo reenvía el paquete a todos sus vecinos si se encuentra más cerca de la zona destino que el nodo del que ha recibido el paquete. De esta manera, el paquete se reenvía cada vez más cerca de la región destino. Finalmente, dentro de la zona destino, el paquete se reenvía a todos los nodos vecinos. Se puede ver un ejemplo gráfico de la aproximación *LBM-step* en la figura 5.

2.5.4 –SIFT

Simple Forwarding over Trajectory, SIFT [10], es un protocolo basado en la posición geográfica de los nodos de la red, pero que a diferencia de otros protocolos de esta categoría que utilizan transmisiones punto a punto, en SIFT el nodo emisor envía el paquete en *broadcast* a todos sus vecinos y es el nodo receptor, en vez de el emisor, el que se encarga de decidir sobre la retransmisión del paquete.

Cuando un nodo recibe un paquete, toma la decisión de reenviar o descartar el paquete en función de su propia situación geográfica y la información contenida en el paquete, que es la localización geográfica del último nodo que envió el paquete y la trayectoria que debe seguir el paquete hacia el destino. Esto supone una gran ventaja, ya que no es necesario que los nodos compartan información de posición entre ellos, es decir, no hay sobrecarga en la red por paquetes de control porque éstos no existen.

Una vez que se recibe el paquete, cada nodo establece un temporizador en función de su posición geográfica con respecto a la trayectoria y el último nodo que reenvió el paquete. De esta forma, el nodo que se encuentre más cerca de la trayectoria y más lejos del último emisor, establece el temporizador al valor más pequeño. Si antes de que el temporizador expire el nodo recibe una copia del mismo paquete, el temporizador se para y el paquete es descartado. En caso contrario, el paquete es retransmitido en *broadcast* cuando el temporizador expira. Así, el nodo con el temporizador más corto, será el que envíe el paquete, es decir, aquel que se encuentre más cerca de la trayectoria y más lejos del último nodo en emitir el paquete.

Además de la información descrita anteriormente, la cabecera del paquete contiene un identificador del nodo origen inicial, un número de secuencia y un contador de los saltos que da el paquete por los nodos. Así, los nodos podrán mantener una lista de los últimos paquetes recibidos para evitar bucles.

El hecho de que los nodos sólo necesiten la información contenida en el paquete para encaminarlo hacia el destino es un dato muy significativo porque hace que el protocolo no precise de intercambio de información de señalización entre nodos y por lo tanto sea una opción interesante para redes con alta tasa de movilidad.

2.5.5 –*GeoGRID*

GeoGRID [11] divide la red en una cuadrícula, de manera que en cada región de la cuadrícula se selecciona como puerta de enlace un nodo que se encuentre cercano al centro. Únicamente las puertas de enlace seleccionadas se encargan de propagar o retransmitir los paquetes a las regiones vecinas, de forma que, los paquetes se envían por la red de cuadrado en cuadrado de la rejilla. De este modo, como sólo las puertas de enlace pueden reenviar paquetes, la sobrecarga por inundación de la red se ve reducida. Además, el tamaño de las regiones de la cuadrícula se ajusta de tal manera que las puertas de enlace puedan comunicarse con al menos una de las puertas de enlace de regiones vecinas.

Existen dos versiones diferentes de *GeoGrid*: *flooding-based GeoGrid* y *ticket-based GeoGrid*.

Flooding-based GeoGrid

En la versión de *GeoGrid* basada en inundación, cuando se envía un paquete, se utiliza una región de reenvío rectangular para reducir aún más la sobrecarga en la red. Así, si una puerta de enlace que se encuentra fuera de la zona de reenvío (rectángulo más pequeño que incluye al emisor del mensaje y al destino) recibe un paquete, éste es descartado. En cambio, si la puerta de enlace se encuentra dentro de la zona de reenvío, ésta reenvía el paquete a sus puertas de enlace vecinas, sólo si el paquete no había sido reenviado anteriormente. Una vez que el paquete llega a la región destino, el paquete se difunde mediante inundación simple.

Ticket-based GeoGrid

En esta versión de *GeoGrid* se sigue haciendo uso de la zona de reenvío, pero sólo un cierto número de puertas de enlace que se encuentren dentro de la zona de reenvío podrán retransmitir el paquete. El emisor del paquete puede controlar el número de puertas de enlace que pueden reenviar el paquete difundiendo un determinado número de *tickets*, de tal manera que un

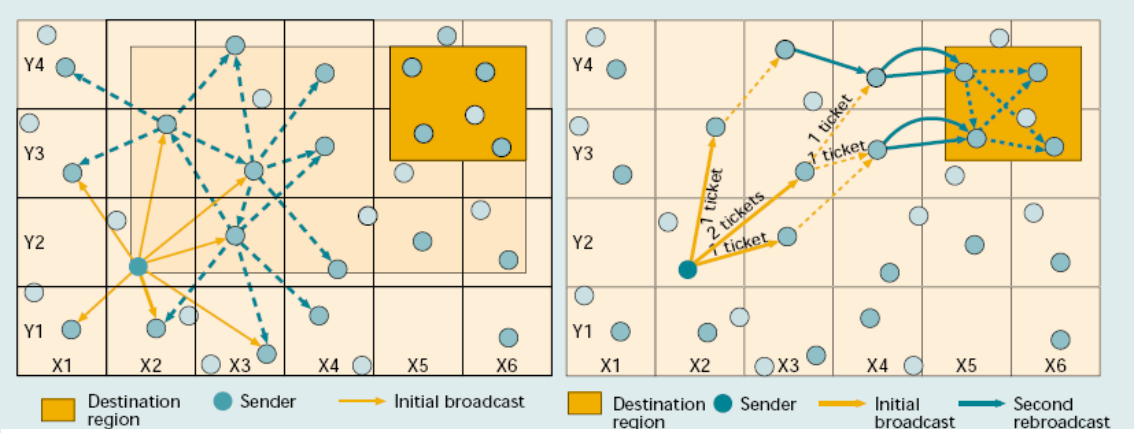

Figura 6 : Ejemplo de *flooding-based* y *ticket-based GeoGrid*

paquete sólo es reenviado si la puerta de enlace está en posesión de un *ticket*. De esta manera, no todas las puertas de enlace que están dentro de la zona de reenvío pueden retransmitir el paquete y el emisor puede controlar tanto la sobrecarga de la red por inundación, como la probabilidad de que el paquete llegue al destino difundiendo mayor o menor número de *tickets*.

Hay que señalar que cada *ticket* representa el envío de una copia del paquete hacia la región destino, así que, si una puerta de enlace recibe dos *tickets*, realizará dos transmisiones del paquete duplicado y no lo descartará.

Cuando el paquete llega a la región destino, el paquete se hace llegar a todos los nodos mediante inundación simple.

En la figura 6 se muestran unos ejemplos sobre ambas versiones de *GeoGrid*.

En la imagen de la izquierda de la figura 6 se muestra la versión de *GeoGrid* basada en inundación. Se puede apreciar como inicialmente se envía el paquete a todas las puertas de enlace vecinas, pero que únicamente las que se encuentran dentro de la zona de reenvío retransmiten el paquete.

A la derecha de la figura 6 se muestra un ejemplo de *GeoGrid* basado en *tickets* donde se puede ver como inicialmente se distribuyen cuatro *tickets* entre las puertas de enlace vecinas y que se van distribuyendo junto con los paquetes hasta llegar a la región destino donde, finalmente, se utiliza inundación simple. Como inicialmente el emisor difunde cuatro *tickets*, llegan cuatro copias del mensaje a la región destino.

2.5.6- *GeoNode*

El encaminamiento llevado a cabo por *GeoNode* [5,12] necesita una red con infraestructura que tendrá una arquitectura celular con un GeoNodo (o MSS, *mobile support station*) por cada célula (semejante a la estructura GSM). Así, el encaminamiento de un paquete se divide en dos fases: primero entre el emisor y el GeoNodo, y después, entre el GeoNodo y la región destino o destinatario. Para llevar a cabo el encaminamiento se utilizan las coordenadas GPS de longitud y latitud terrestres.

Se presentan tres soluciones para integrar las direcciones basadas en la localización geográfica en el diseño de internet:

- Solución a nivel de aplicación extendiendo los servicios DNS actuales

  Se trata de extender DNS incluyendo entradas en los servidores con información de localización geográfica. Se incluiría un nuevo dominio de primer nivel ".geo" para este propósito. Por debajo de este nivel, el segundo nivel representaría a países, el tercero a ciudades y por último, el cuarto nivel, representaría polígonos de coordenadas geográficas. De esta forma, una consulta de una localización geográfica nos devolvería un conjunto de direcciones IP de los GeoNodos que cubren la región destino. Así, el paquete se podría enviar a cada uno de los GeoNodos y después, ser retransmitido por los GeoNodos dentro del área destino.

- GPS-*Multicast*

  La precisión del encaminamiento geográfico se ve limitado por el número de direcciones IP disponibles, sobre todo con IPv4, ya que no existirían suficientes direcciones IP para hacer las regiones geográficas tan pequeñas como se quisiera. De esta manera, se definen las regiones geográficas lo más pequeñas posible como para ser direccionables, se las llama átomos. Así, un área geográfica más o menos extensa estará formada por un conjunto de átomos.

  Cada átomo y conjunto de átomos que forman una región se asocia a una dirección *multicast* que es utilizada para encaminar los paquetes del emisor al GeoNodo que cubre la región destino. Así, cada GeoNodo debe unirse a los grupos *multicast* de las regiones geográficas que cubre.

  Cuando el emisor desea enviar un paquete a una determinada región geográfica utilizará la dirección *multicast* del conjunto de átomos más pequeño que cubre el área donde desea enviar el paquete. En el paquete enviado figurará como dirección destino la dirección *multicast*, con lo que el GeoNodo asociado a la región destino recibirá el paquete. Además, se incluirá en el paquete información sobre el área destino para que en la segunda fase del encaminamiento entre el GeoNodo y el destino, el paquete se entregue dentro del área deseada.

- Encaminamiento IP *unicast* extendido para tratar con direcciones GPS

  Para poder integrar la localización geográfica dentro de las decisiones de encaminamiento y poder realizar encaminamiento geográfico son necesarios tres componentes en la red:

  GeoRouters: se encargan de transportar los paquetes del origen al destino. Para realizarlo intercambian entre ellos información sobre su área de servicio. Para mejorar la eficiencia se organizan jerárquicamente de forma que los *georouters* de menor nivel jerárquico u hojas prestan servicio a una pequeña área geográfica y los *georouters* que se encuentran en niveles de jerarquía superiores engloban las áreas geográficas de los *georouters* que tienen por debajo.

GeoNodos: su función es almacenar los paquetes dirigidos a su área geográfica de influencia durante su tiempo de vida y retransmitirlos periódicamente mediante *multicast* dentro de su celda asociada.

GeoHost: simplemente es un demonio que se encuentra en ejecución en todos los *host* que se encarga de recibir y enviar paquetes. Además, proporciona información a los demás procesos de la llegada de nuevos paquetes, situación geográfica actual y dirección del GeoNodo.

Para enviar un paquete, el emisor necesita conocer la dirección IP del GeoNodo de su celda. Entonces, el paquete es enviado al GeoNodo y éste lo retransmite a su *GeoRouter* local. Una vez recibido el paquete por el *GeoRouter*, éste determina si la región destino a la que va dirigido el paquete interseca con su área de influencia. Si la comprobación es negativa, el paquete se envía al *GeoRouter* padre. Si el *GeoRouter* sólo cubre parcialmente la región destino del paquete, se le retransmite igualmente al *GeoRouter* padre una copia del paquete. En el caso de que la región destino del paquete y el área de influencia del *GeoRouter* intersequen, el *GeoRouter* envía una copia del paquete a los hijos cuya área de servicio interseque con la región destino del paquete. Finalmente, los *GeoRouters* entregan los paquetes a los GeoNodos correspondientes y éstos difunden el paquete dentro de la región destino.

En la última fase entre el GeoNodo y el destino, el encaminamiento puede estar basado en filtrado de los paquetes a nivel de aplicación o filtrado *multicast*.

Con filtrado a nivel de aplicación, el GeoNodo utiliza una o varias direcciones *multicast* para transmitir el paquete dentro de la celda. Entonces, cada nodo comprobará a nivel de aplicación si su situación geográfica pertenece o no al área destino (que está incluida en el paquete), aceptando o descartando el paquete en consecuencia.

Con filtrado *multicast*, la comprobación se realiza a nivel IP. El GeoNodo envía, utilizando una dirección *multicast* previamente conocida por todos los nodos de la celda, una lista con los paquetes disponibles, sus regiones destino y su grupo *multicast* temporal. Así, los nodos se unirán a los grupos *multicast* en los que estén interesados y donde posteriormente el GeoNodo enviará los paquetes anunciados.

En la figura 7 se muestra un ejemplo. El emisor transmite el paquete a su GeoNodo local GN1 que lo reenvía a su *GeoRouter* GR1. Como el área de influencia de GR1 no interseca con la región destino, éste retransmite el paquete al *GeoRouter* GR2. El área de servicio de GR2 incluye a la región destino, por lo que se envía el paquete su hijo, el *GeoRouter* GR3. Posteriormente, GR3 transmite el paquete a los GeoNodos GN2 y GN3 cuyas celdas intersecan con la región destino del paquete. Finalmente, se difunde el paquete dentro de la celda con *multicast*.

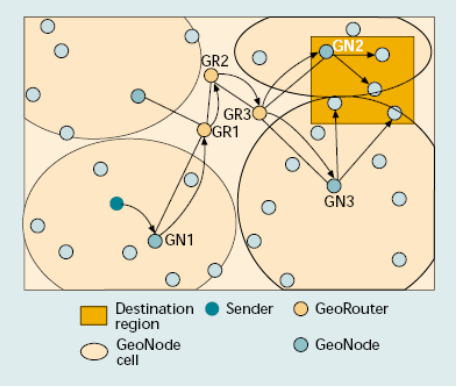

Figura 7 : Ejemplo de encaminamiento en *GeoNode*

2.5.7 – GPSR

GPSR, *Greedy Perimeter Stateless Routing* [13], utiliza dos formas de transmitir paquetes: *greedy forwarding*, que se utiliza siempre que es posible, y *perimeter forwarding* que se utiliza en los casos en los que no se puede hacer uso de *greedy forwarding*.

Greedy forwarding

La idea es que si un nodo conoce la posición geográfica de sus nodos vecinos, éste puede determinar el mejor vecino al que enviar el paquete, que será aquel que se encuentre más cerca de la posición geográfica del destino. De este modo, cuando un emisor envía un paquete, éste es enviado de vecino en vecino siguiendo la premisa anterior hasta que se llega al destino.

Un ejemplo de elección del mejor vecino se muestra en la figura 8, donde se aprecia como el nodo 'x' decide enviar el paquete al nodo 'y' por ser el más cercano al destino D entre los vecinos que se encuentran en su radio de alcance. Posteriormente el nodo 'y' reenviará el paquete a su vecino más cercano al destino y se procederá de la misma forma, cada nodo retransmitiendo el paquete al vecino más cercano al destino, hasta que el paquete llegue a D.

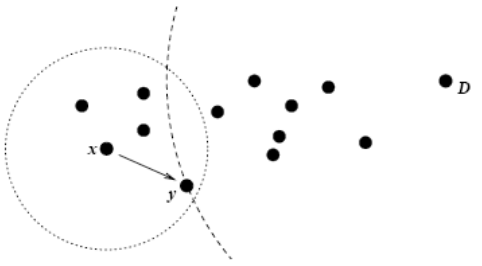

Figura 8 : Ejemplo de elección de vecino GPSR

Para que los nodos tengan conciencia de la localización geográfica de sus vecinos se utiliza un algoritmo de balizas. Cada nodo transmite periódicamente una baliza que contiene su identificador y su posición hacia el resto de vecinos. Cuando un nodo recibe una baliza, éste almacena la información que contiene para posibles usos futuros en una tabla de localización.

Las balizas deben ser enviadas con periodo T ya que cuando un nodo detecta que expira el tiempo de una baliza considera que el vecino es inaccesible y elimina la información oportuna de las tablas. La elección del periodo T debe ser adecuada puesto que desde la última baliza recibida pueden haber salido o entrado nuevos nodos del rango de alcance. Por lo tanto, el periodo T dependerá de la movilidad de la red y del rango de alcance de los nodos.

Para minimizar la sobrecarga en la red provocada por las balizas se utiliza *piggybacking*, de tal forma que cuando un nodo transmite un paquete de datos, adjunta en el paquete su posición. Dicha información será recibida por todos los nodos que se encuentren dentro del rango de alcance de emisión, ya que se reciben paquetes en modo promiscuo. Además, se puede minimizar aún más la sobrecarga debida a las balizas utilizando unos mensajes de petición, de manera que un nodo sólo solicitará balizas a los vecinos cuando necesite enviar algún paquete de datos.

El esquema de *greedy forwarding* presenta problemas con topologías como las que se muestra en la figura 9, donde el nodo x no puede encontrar ningún vecino dentro de su radio de alcance que se encuentre más cerca del destino D que él mismo. En estas situaciones se debe enviar el paquete temporalmente más lejos del destino y es donde se utiliza *perimeter forwarding.*

Perimeter forwarding

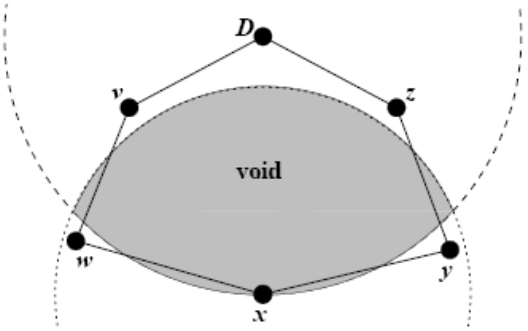

Figura 9 : Ejemplo de utilización de *perimeter forwarding*

El mecanismo de *perimeter forwarding* se utiliza cuando falla el de *greedy forwarding*. Éste resulta de utilizar el concepto de la regla de la mano derecha. De forma resumida, la regla de la mano derecha viene a decir que una persona encerrada en un laberinto sólo tiene que caminar con su mano derecha pegada a la pared para recorrer todos los muros del mismo y que, por consiguiente, en algún momento encuentra la salida. En lugar de un laberinto, modelando una red como un grafo, lo que tenemos es un conjunto de vértices que representa los nodos de la red y un conjunto de aristas que representa la conectividad directa entre nodos, es decir, la posibilidad de enviar un mensaje directamente y que éste sea recibido correctamente. En este caso, la regla de la mano derecha se puede utilizar para recorrer el perímetro de de la zona sin vecinos (*void* en la figura 9) hasta llegar a un nodo que se encuentre más cerca del destino que el nodo inicial x.

Por lo tanto, GPSR utiliza siempre que es posible *greedy forwarding* salvo en aquellos casos mencionados anteriormente en los que es necesario aplicar *perimeter forwarding*.

Puede darse el caso de que el destino D no sea alcanzable porque no se encuentre dentro del rango de alcance de ningún nodo, en tal caso, esto será detectado por GPSR y se descartará el paquete.

Como conclusión señalar que GPSR es un algoritmo de encaminamiento geográfico que utiliza información de localización geográfica para conseguir que la información de estado en los nodos sea muy pequeña (de ahí proviene el calificativo *stateless* del protocolo). También decir, que la complejidad de los mensajes del protocolo de encaminamiento es muy pequeña gracias a que sólo es necesaria la localización geográfica de los vecinos dentro del rango de alcance para encaminar los paquetes. Además, GPSR es robusto en la entrega de paquetes en redes vehiculares suficientemente densas.

Capítulo 3

Diseño del protocolo de encaminamiento geográfico y de la integración con IPv6

Para la elección del protocolo de encaminamiento geográfico que se ha implementado nos hemos guiado por las soluciones en las que se están trabajando en proyectos europeos y diversos organismos de estandarización, como el *Car 2 Car Communication Consortium* [4] o el proyecto *Geonet* [14] que se basan en la utilización de GPSR para encaminar los paquetes de envíos *geounicast* e inundación simple para la distribución de un envío *geobroadcast* dentro de su área geográfica destino. En concreto, se han seguido las especificaciones del documento "GeoNet D2.1 Specifications- 1st realease" [15] del proyecto *Geonet*. Del mismo modo, para la integración de IPv6 sobre el protocolo de encaminamiento geográfico se han seguido las especificaciones de [15] y [16].

3.1- Protocolo de encaminamiento geográfico

En el protocolo de encaminamiento geográfico implementado se definen dos tipos de envío de paquetes, entrega de paquetes *geounicast* y entrega de paquetes *geobroadcast*.

Para la entrega de paquetes *geounicast* nos basamos en GPSR para encaminar los paquetes. Recordemos que GPSR utilizaba dos tipos de encaminamiento, *greedy forwarding*, que se basaba únicamente en la posición geográfica de los nodos vecinos para tomar las decisiones de encaminamiento, y *perimeter forwarding*, que se utilizaba para recuperarse cuando fallaba *greedy forwarding*, encaminando los paquetes por el perímetro en busca de un nodo que se encuentre más cerca del destino. En la implementación únicamente realizamos el encaminamiento con *greedy forwarding*. El motivo por el que no se implementa el encaminamiento *perimeter forwarding* es porque su complejidad es elevada y en circunstancias no ideales, no resuelve eficientemente todos los casos en los que falla *greedy forwarding*, sino que realiza una búsqueda exhaustiva por toda la red [17].

Para la entrega de paquetes *geobroadcast*, se utiliza *greedy forwarding* de GPSR para encaminar los paquetes hacia la región geográfica destino e inundación simple para hacer llegar el paquete a todos los nodos dentro de dicha zona.

En ambos tipos de envío, el encaminamiento de los paquetes se basa en la posición geográfica que ocupan los nodos de la red. Para ello, todos los nodos necesitan saber la posición geográfica

que tienen sus nodos vecinos (aquellos que se encuentran en contacto directo, es decir, dentro del rango de cobertura).

Para que los nodos puedan determinar la localización geográfica de sus vecinos se utiliza un algoritmo de balizas o *beacon*. En este algoritmo, cada nodo transmite periódicamente un paquete baliza con su posición geográfica al resto de nodos vecinos. Cuando un nodo recibe una baliza, éste almacena la información que contiene en una tabla, la tabla de vecinos.

Las balizas deben ser enviadas cada cierto periodo de tiempo, ya que, cuando un nodo detecta que expira el tiempo de validez de una entrada en la tabla de vecinos, se considera que el vecino es inaccesible y se elimina de la tabla. El periodo de envío de balizas es un parámetro significativo puesto que, desde la última baliza recibida, pueden haber salido o entrado nuevos nodos del rango de alcance. Por lo tanto, este periodo dependerá de la movilidad de la red y el rango de cobertura de los nodos. En la implementación, el periodo de envío de balizas es de 500 milisegundos y el tiempo de validez de una entrada en la tabla de vecinos es de 10 segundos. Se han adoptado estos valores siguiendo las especificaciones definidas en el documento del proyecto *Geonet* [15], aunque como trabajo futuro, se podrían plantear estudios sobre el efecto de la variación de estos tiempos.

Para minimizar la sobrecarga en la red provocada por las balizas se utiliza *piggybacking*. De esta forma, cuando un nodo transmite un paquete de datos, adjunta en el paquete su posición geográfica tal y como si se tratara de un paquete baliza.

3.1.1- Entrega de paquetes *Geounicast*

Un envío *geounicast* se define como la entrega de un paquete a un único nodo que se encuentra en una determinada posición geográfica. El encaminamiento del paquete se realiza en función de la posición geográfica en la que se encuentra el destino. Se trata de una comunicación punto a punto.

El comportamiento de un nodo para un envío *geounicast* será el siguiente:

- Si se genera un paquete de datos para enviar (se recibe un paquete del nivel superior de la pila de protocolos, en nuestro caso IPv6) o se recibe un paquete de otro nodo vecino se comprueba, en primer lugar, si el nodo es el destinatario del paquete.

- En caso afirmativo, el nodo es el destino del paquete y éste se transfiere al nivel superior de la pila de protocolos, en nuestro caso IPv6.

- En caso negativo, se consulta en la tabla de nodos vecinos (aquellos que se encuentran en contacto directo con él, es decir, dentro de su rango de cobertura) la posición geográfica de cada uno de ellos y selecciona el mejor vecino al que reenviar el paquete.

- El mejor vecino al que reenviar el paquete será el destinatario del mismo, en el caso de que éste se encuentre dentro de la lista de vecinos o, en su ausencia, aquel que se encuentra más cerca de la posición geográfica del destino, que está incluida en el paquete.

Hay que señalar que los nodos pueden determinar la localización geográfica de sus vecinos por la utilización del algoritmo de balizas.

Si todos los nodos de la red implementan este comportamiento, el resultado es que el paquete se reenvía de nodo en nodo hasta el destino, siempre y cuando la densidad de nodos en la red sea suficientemente alta como para que exista conectividad entre emisor y destino por medio de un puente multisalto.

Un ejemplo de envío *geounicast* se puede ver en la figura 10 [3]. En esta figura, que presenta un escenario de una carretera, se puede apreciar como el vehículo S envía un paquete que a través de una comunicación multisalto llega al destino D. Los vehículos intermedios F, que permiten el encaminamiento del paquete, son elegidos en cada salto como el vecino más cercano a la posición geográfica del destino y se encargan de reenviar el paquete.

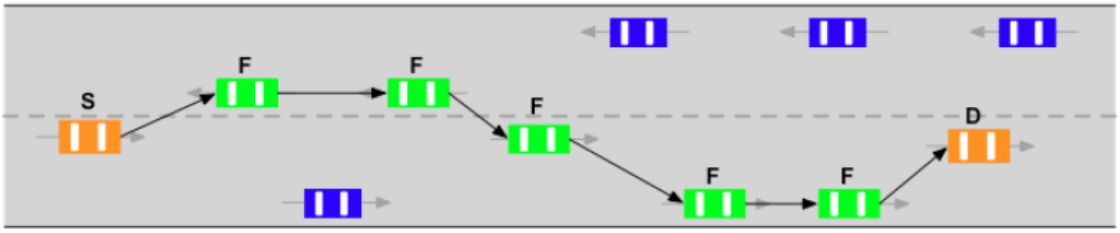

Figura 10 : Ejemplo de envío *geounicast*

3.1.2- Entrega de paquetes *Geobroadcast*

Un envío *geobroadcast* se define como la entrega de un paquete a todos los nodos que se encuentran dentro de una determinada región geográfica. El encaminamiento del paquete se realiza en función de la posición geográfica del área destino.

El comportamiento de un nodo para un envío *geobroadcast* es similar al de un envío *geounicast* pero con algunas diferencias. Su forma de actuar difiere según se encuentre dentro de la zona destino del paquete o no. Su comportamiento será el siguiente:

- Si se genera un paquete de datos para enviar (se recibe un paquete del nivel superior de la pila de protocolos, en nuestro caso IPv6) o se recibe un paquete de otro nodo vecino se comprueba, en primer lugar, si el nodo se encuentra dentro del área geográfica a la que va dirigida el paquete.

- En caso negativo, el comportamiento es análogo al caso *geounicast*, se consulta en la tabla de nodos vecinos la posición geográfica de cada uno de ellos y selecciona el mejor vecino al que reenviar el paquete. El mejor vecino al que reenviar el paquete es aquel que se encuentra más cerca de la posición geográfica del área destino, que está incluida en el paquete.

- En caso afirmativo, nos encontramos dentro del área geográfica destino y el nodo es uno de los destinatarios, por lo que se transferirá el paquete al nivel superior de la pila de protocolos, en nuestro caso IPv6.

- Por otro lado, para asegurar que el paquete llega a todos los nodos que se encuentran dentro del área geográfica destino, se debe recurrir a inundación simple, pero únicamente dentro del área destino del paquete. De esta manera, cuando un nodo determina que se encuentra dentro del área geográfica destino, comprueba si es la primera vez que recibe el paquete *geobroadcast*, consultando una tabla de paquetes *geobroadcast* recibidos anteriormente.

- Si la comprobación es positiva y es la primera vez que se recibe el paquete *geobroadcast*, además de transferirlo al nivel superior, se retransmite una copia a todos los nodos vecinos para garantizar que paquete llega a todos los nodos que se encuentran dentro del área geográfica destino.

- Si por el contrario, el paquete *geobroadcast* ya ha sido recibido anteriormente, éste se descarta para evitar bucles y para que la inundación cese.

Al igual que en la entrega *geounicast*, los nodos pueden determinar la localización geográfica de sus vecinos por la utilización del algoritmo de balizas.

De esta manera, el resultado es que el paquete *geobroadcast* se reenvía de nodo en nodo hasta un nodo que se encuentra dentro del área geográfica destino como si se tratara de un envío *geounicast*, y una vez que se llega al área geográfica destino, el paquete se distribuye a todos los nodos dentro de la región destino mediante inundación simple.

Un ejemplo de envío *geobroadcast* se puede ver en la figura 11 [3]. En esta figura, que presenta un escenario de una carretera, se puede apreciar como el vehículo S envía un paquete hacia todos los nodos que se encuentran dentro de un área geográfica destino. Los vehículos intermedios, F, que permiten el encaminamiento del paquete, son elegidos en cada salto como el vecino más cercano a la región geográfica del destino y se encargan de reenviar el paquete hasta el vehículo F/D que se encuentra dentro de la región destino. Posteriormente, se utiliza inundación simple para hacer llegar el paquete *geobroadcast* a todos los vehículos D dentro del área destino.

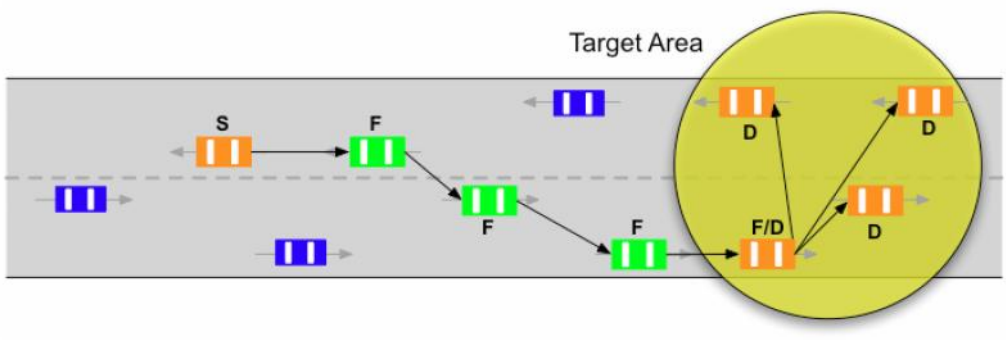

Figura 11 : Ejemplo de envío *geobroadcast*

Por último señalar que se permite definir el área destino de un paquete *geobroadcast* como un círculo o como un rectángulo, aunque la forma de la región destino no influye en el comportamiento del protocolo.

3.2- Cabeceras y tipos de paquetes del nivel de encaminamiento geográfico

El diseño de los tipos de paquetes y sus cabeceras del nivel de encaminamiento geográfico se ha obtenido de las especificaciones del documento público del proyecto *Geonet* [15]. De estas especificaciones se han seleccionado aquellos tipos de paquetes y campos de sus cabeceras que eran más adecuados para llevar a cabo la implementación del protocolo de encaminamiento geográfico planteado.

Los paquetes siguen el esquema de la figura 12. Como se puede ver en ella, la cabecera de nivel de encaminamiento geográfico está compuesta por una cabecera común a todos los tipos de paquete y por otra parte dependiente del tipo de paquete.

En los siguientes apartados se muestran los tipos de paquetes y cabeceras resultantes.

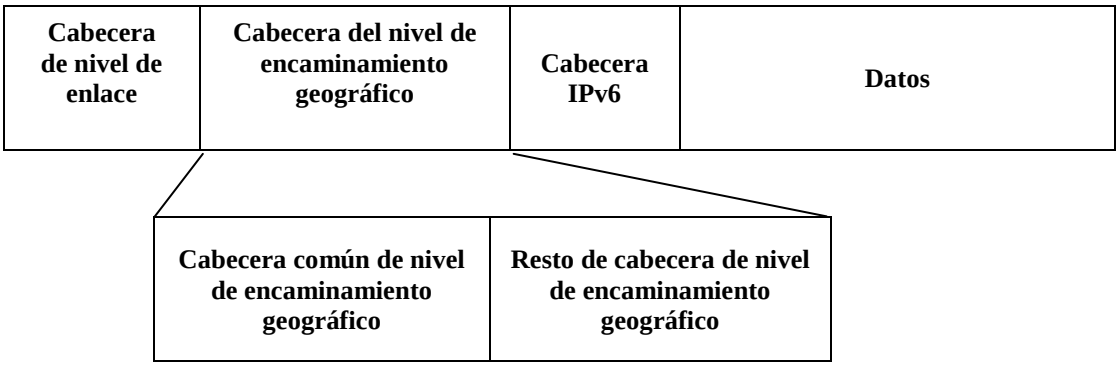

Figura 12 : Estructura de un paquete

3.2.1- Definición de tipos de paquetes

La cabecera común del nivel de encaminamiento geográfico tiene un campo en el que se indica el tipo y subtipo de paquete.

En la tabla 1 se indican los tipos de paquete que se distinguen a nivel de encaminamiento geográfico:

Tipo de paquete	Valor del campo en la cabecera común	Descripción
Beacon	1	Paquete baliza
Unicast	2	Paquete de envío *geounicast*
Broadcast	4	Paquete de envío *geobroadcast*

Tabla 1 : Tipos de paquetes

En la tabla 2 se indican los subtipos de los paquetes que existen a nivel de encaminamiento geográfico:

Subtipo de paquete	Valor del campo en la cabecera común	Descripción
GeoBroadcast_circle_subtype	0	Señala que la zona destino del paquete *geobroadcast* se define como un círculo
GeoBroadcast_rect_subtype	1	Señala que la zona destino del paquete *geobroadcast* se define como un rectángulo

Tabla 2 : Subtipos de paquetes

3.2.2- Descripción del formato de los paquetes

En este apartado se presenta el formato de los paquetes del nivel de encaminamiento geográfico y se describe el significado de cada campo de las cabeceras.

3.2.2.1- Cabecera común del nivel de encaminamiento geográfico

La cabecera común del nivel de encaminamiento geográfico es común a todos los paquetes independientemente de su tipo. Está formada por los siguientes campos:

- *Version* (Versión de protocolo): identifica la versión del protocolo de nivel de encaminamiento geográfico.

- *Prio* (Prioridad): se trata de un entero sin signo que indica la prioridad del paquete. Este campo, aunque aparece en nuestra implementación del protocolo de encaminamiento geográfico, no repercute en el comportamiento del mismo. De todas formas, sí que se le puede asignar un valor concreto para posibles ampliaciones de funcionamiento futuras.

- *HT* (Tipo de cabecera): indica el tipo de paquete o cabecera de nivel de encaminamiento geográfico. Los valores que puede tomar este campo y los tipos de paquete se han definido en la tabla 1.

- *HST* (Subtipo de cabecera): indica el subtipo de paquete o cabecera de nivel de encaminamiento geográfico. Los valores que puede tomar este campo y los subtipos de paquete se han definido en la tabla 2.

- *TxPower* (Potencia de transmisión): es un entero sin signo que indica el nivel de potencia de transmisión con el que el paquete fue enviado. Se expresa en ½ de dBm. Por ejemplo, para una potencia de transmisión de 10 dBm, el campo toma un valor de

20. En nuestra implementación el valor de este campo es irrelevante, aunque sí que se le puede asignar un valor concreto para posibles ampliaciones de funcionamiento futuras.

- *Reserved (Flags)*: es un campo de uso reservado para posibles ampliaciones futuras. En nuestra implementación este campo toma valor 0.

- *Payload Length* (Longitud de los datos): se trata de un entero sin signo que indica la longitud de los datos del paquete, es decir, el resto del paquete que sigue a la cabecera de nivel de encaminamiento geográfico completa. Se expresa en octetos.

- *Next Header* (Cabecera siguiente): identifica el tipo de cabecera que sigue a la cabecera de nivel de encaminamiento geográfico. En nuestro caso será la cabecera de IPv6, pero puede ser la cabecera de cualquier otro protocolo.

- *Hop Limit* (Límite de saltos): es un entero sin signo que indica el número máximo de saltos (de nodo a nodo de la red) que puede dar un paquete. Este campo, por lo tanto, será reducido en una unidad cada vez que un nodo reenvíe el paquete. Si el valor de este campo llega a 0, el paquete no se retransmite.

- Sección *Position Vector* (Vector de posición): se trata de un conjunto de campos que describen los datos de posición geográfica del nodo que emite el paquete.

La sección *Position Vector* (Vector de posición) está formada por los siguientes campos:

- *Source ID* (Identificador de fuente): se trata de una dirección de 64 bits. Es el identificador de nivel de encaminamiento geográfico del nodo que retransmite el paquete.

- *Source TS* (Marca de tiempo de la fuente): es un entero de 32 bits sin signo que expresa el instante de tiempo en milisegundos en el que el nodo fuente obtuvo su posición geográfica, es decir, su latitud y longitud. El tiempo se expresa como el número de milisegundos desde las 0:00 horas del 1 de Enero de 1970 (*Unix Epoch midnight*).

- *Source Latitude* (Latitud de fuente): es un entero de 32 bits con signo que indica la latitud del nodo que genera la cabecera de nivel de encaminamiento geográfico. Está expresada en unidades de 1/8 de microgrado.

- *Source Longitude* (Longitud de fuente): es un entero de 32 bits con signo que indica la longitud del nodo que genera la cabecera de nivel de encaminamiento geográfico. Está expresada en unidades de 1/8 de microgrado.

- *Source Speed* (Velocidad de fuente): es un entero de 16 bits con signo que indica la velocidad del nodo que genera la cabecera de nivel de encaminamiento geográfico. Está expresado en unidades de 0,01 metros por segundo.

- *Source Heading* (Dirección de Fuente): es un entero de 16 bits sin signo que indica la dirección del nodo que genera la cabecera de nivel de encaminamiento geográfico. Está expresado en unidades de 0.005493247 grados, tomando como referencia el Norte.

- *Source Altitude* (Altitud de fuente): es un entero de 16 bits con signo que indica la altitud a la que se encuentra el nodo que genera la cabecera de nivel de encaminamiento geográfico. Está expresado en unidades de metros.

- *TimAc* (Precisión temporal): es un indicador de la precisión del valor expresado en el campo *Source TS*. En nuestra implementación el valor de este campo es irrelevante, aunque sí que se le puede asignar un valor concreto para posibles ampliaciones de funcionamiento futuras.

- *PosAc* (Precisión de posición): es un indicador de la precisión de la posición dada en los campos *Source Latitude* y *Source Longitude*. En la implementación realizada el valor de este campo es irrelevante, aunque sí que se le puede asignar un valor concreto para posibles ampliaciones de funcionamiento futuras.

- *SAc* (Precisión de velocidad): es un indicador para la precisión del valor expresado en el campo *Source Speed*. Al igual que con los otros campos de precisión, en la implementación realizada el valor de este campo es irrelevante, aunque sí que se le puede asignar un valor concreto para posibles ampliaciones de funcionamiento futuras.

- *HAc* (Precisión de dirección): se trata de un indicador de la precisión del valor indicado en el campo *Source Heading*. En la implementación realizada el valor de este campo es irrelevante, aunque sí que se le puede asignar un valor concreto para posibles ampliaciones de funcionamiento futuras.

- *AAc* (Precisión de altitud): se trata de un indicador de la precisión del valor indicado en el campo *Source Altitude*. En la implementación realizada el valor de este campo es irrelevante, aunque sí que se le puede asignar un valor concreto para posibles ampliaciones de funcionamiento futuras.

Por otro lado hay que señalar que la cabecera común del nivel de encaminamiento geográfico se modifica en cada salto de nodo a nodo. Por lo tanto todos los valores de posición, velocidad, dirección, altitud, precisión, etc. de la sección *Position Vector* de la cabecera común, se refieren al último nodo que reenvió el paquete y no al nodo origen que lo transmitió.

También hay que destacar que la cabecera común está en consonancia con los trabajos que se están realizando en el C2C-CC [4].

3.2.2.2- Paquete baliza o beacon

El paquete baliza se utiliza para informar a los nodos vecinos, aquellos que se encuentran dentro del rango de cobertura, de la posición del nodo emisor. Tal y como se muestra en la figura 13, el paquete baliza únicamente está compuesto por la cabecera común de

nivel de encaminamiento geográfico, ya que en ésta se encuentran todos los datos necesarios para su cometido.

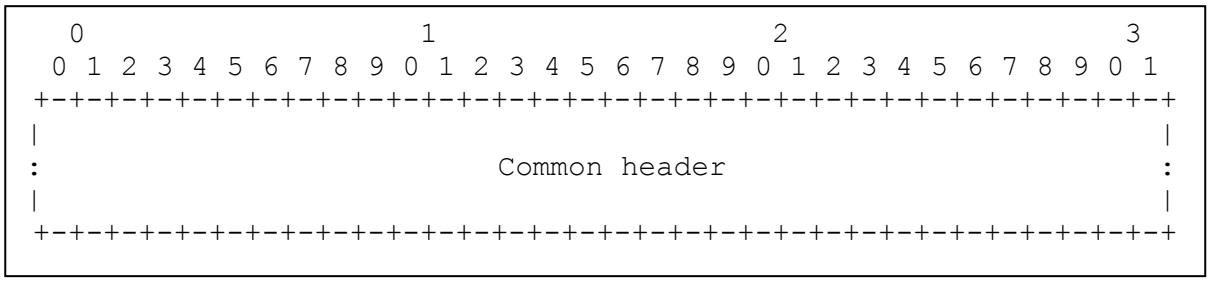

Figura 13 : Representación ASCII del paquete baliza

Hay que señalar que algunos de los campos de esta cabecera común tienen un valor fijo por tratarse de un paquete baliza:

HT (Tipo de cabecera): toma el valor del tipo de paquete *beacon* que es igual 1.

HST (Subtipo de cabecera): toma el valor 0 al tratarse de un tipo de paquete *beacon*.

Payload Length (Longitud de los datos): toma el valor 0 ya que no hay datos detrás de la cabecera de nivel de encaminamiento geográfico.

Hop Limit (Límite de saltos): se establece el número máximo de saltos a 1 puesto que sólo se informa de la posición a los nodos vecinos que se encuentran dentro del rango de cobertura.

3.2.2.3- Paquete Geounicast

El paquete *geounicast* se utiliza para realizar un envío *geounicast.* Su formato es el de la figura 14.

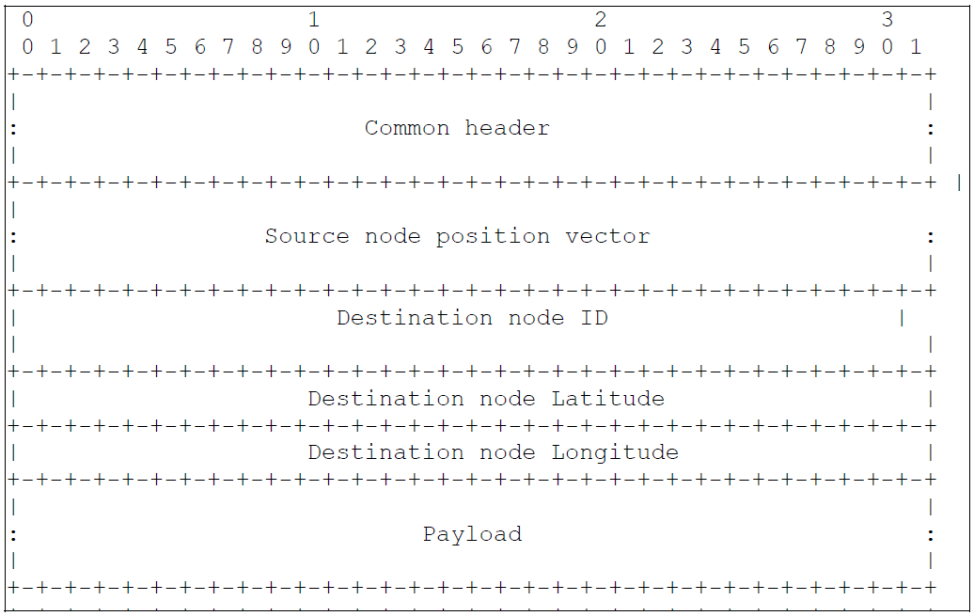

Figura 14 : Representación ASCII del paquete *geounicast*

A continuación vemos cada uno de los campos y secciones que forman esta cabecera:

- *Common Header* (Cabecera Común): es la cabecera común a todos los tipos de paquete. Está formada, entre otras cosas, por los datos de la posición del nodo que retransmite el paquete. Sus campos se han descrito en el apartado 3.2.2.1.

- *Source Node Position Vector* (Vector de posición del nodo fuente): Sus campos son análogos a los de la sección *Position Vector* de la cabecera común del nivel de encaminamiento geográfico que se ha descrito en el apartado 3.2.2.1. La diferencia es que todos los valores de posición, velocidad, dirección, altitud, precisión, etc. se refieren al nodo origen del paquete y no al último nodo que lo reenvió.

- *Destination node ID* (Identificador del nodo destino): se trata de una dirección de 64 bits. Es el identificador de nivel de encaminamiento geográfico del nodo al que va dirigido el paquete.

- *Destination node Latitude* (Latitud del nodo destino): es un entero de 32 bits con signo que indica la latitud del nodo al que va dirigido el paquete. Está expresada en unidades de 1/8 de microgrado.

- *Destination node Longitude* (Longitud del nodo destino): es un entero de 32 bits con signo que indica la longitud del al que va dirigido el paquete. Está expresada en unidades de 1/8 de microgrado.

- *Payload* (Datos): son los datos del paquete. En la implementación, los datos encapsulados corresponden al paquete IPv6 generado por el nodo origen.

Algunos campos de la cabecera común de nivel de encaminamiento geográfico toman un valor fijo por tratarse de un paquete *geounicast*:

HT (Tipo de cabecera): toma el valor del tipo de paquete *geounicast* que es igual 2.

HST (Subtipo de cabecera): toma el valor 0 al tratarse de un tipo de paquete *geounicast*.

3.2.2.4- Paquete Geobroadcast

El paquete *geobroadcast* se utiliza para realizar un envío *geobroadcast*. Su formato aparece en la figura 15.

A continuación vemos cada uno de los campos y secciones que forman esta cabecera:

- *Common Header* (Cabecera Común): es la cabecera común a todos los tipos de paquete. Está formada, entre otras cosas, por los datos de la posición del nodo que retransmite el paquete. Sus campos se han descrito en el apartado 3.2.2.1.

- *Source Node Position Vector* (Vector de posición del nodo fuente): Su división en campos es idéntica a la sección *Position Vector* de la cabecera común del nivel de encaminamiento geográfico que se ha descrito en el apartado 3.2.2.1. La diferencia aquí

es que todos los valores de posición, velocidad, dirección, altitud, precisión, etc. se refieren al nodo origen del paquete y no al último nodo que lo reenvió.

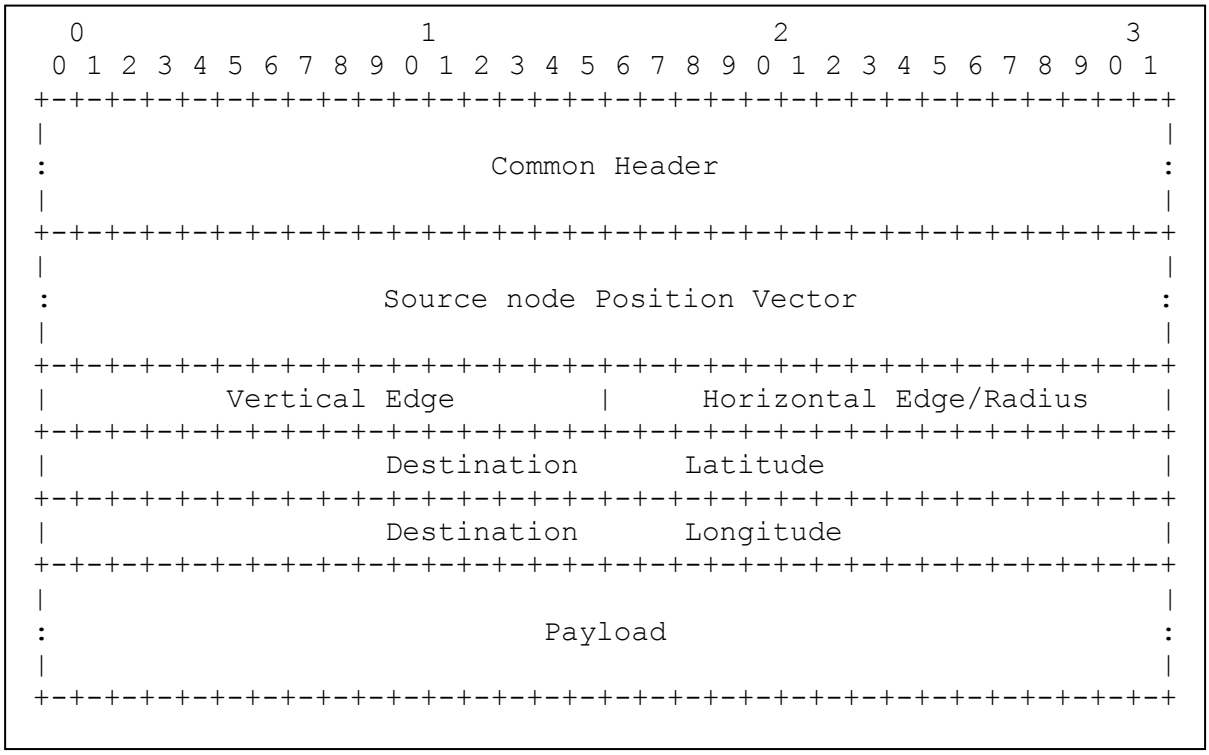

Figura 15 : Representación ASCII del paquete *geobroadcast*

- *Vertical Edge* (Arista vertical): se trata de un entero de 16 bits sin signo. Este campo sólo tiene sentido cuando la región destino del paquete es un rectángulo, es decir cuando el campo de la cabecera común *HST* es igual a 1 (*GeoBroadcast_rect_subtype*). Este campo representa la longitud en metros de la arista vertical del rectángulo que define la región destino, entendiendo por arista vertical, aquella que se orienta de norte a sur. Si el área destino del paquete es un círculo, este campo toma el valor 0.

- *Horizontal Edge/Radius* (Arista horizontal/Radio): se trata de un entero de 16 bits sin signo. Este campo tiene un significado diferente en función de la forma de la región destino del paquete. Si la región destino es un rectángulo, es decir cuando el campo de la cabecera común *HST* es igual a 1 (*GeoBroadcast_rect_subtype*), este campo representa la longitud en metros de la arista horizontal del rectángulo que define la región destino, entendiendo por arista horizontal, aquella que se orienta de este a oeste. Si por el contrario, el área destino del paquete es un círculo, es decir cuando el campo de la cabecera común *HST* es igual a 0 (*GeoBroadcast_circle_subtype*), este campo representa la longitud en metros del radio del círculo que define la zona destino del paquete.

- *Destination Latitude* (Latitud del destino): es un entero de 32 bits con signo que indica la latitud del centro de la zona geográfica destino del paquete. Está expresada en unidades de 1/8 de microgrado.

- *Destination Longitude* (Longitud del destino): es un entero de 32 bits con signo que indica la longitud del centro de la zona geográfica destino del paquete. Está expresada en unidades de 1/8 de microgrado.

- *Payload* (Datos): son los datos del paquete. En la implementación, los datos encapsulados corresponden al paquete IPv6 generado por el nodo origen.

Hay que señalar que el campo *HT* de la cabecera común de nivel de encaminamiento geográfico toma un valor fijo igual a 4 por tratarse de un paquete *geobroadcast*.

3.3- Especificación de IPv6 sobre el nivel de encaminamiento geográfico

Para la integración de IPv6 sobre el nivel de encaminamiento geográfico se han seguido las especificaciones de los documentos [15] y [16]. De esta manera se consigue que los nodos puedan soportar IPv6 y así poder comunicarse con otros nodos de la red vehicular o *host* de internet utilizando IPv6.

Antes de proceder a describir la integración de IPv6 y el nivel de encaminamiento geográfico conviene recordar el concepto de OBU (*On-Board Unit*) y RSU (*Road Side Unit*) vistos en el apartado 2.4.1. Una OBU es una unidad de a bordo que dota a las unidades de aplicación (AU-*Aplication Units*) de los vehículos de capacidades de comunicación con otras OBUs o con las RSUs. Las RSUs pueden verse como nodos fijos de la red VANET que tienen la función básica de enviar, recibir y retransmitir paquetes para aumentar el rango de cobertura de la red *ad hoc*. También pueden ofrecer acceso a internet al encontrarse conectadas a la red fija de algún operador. De esta manera, las RSUs permiten a los vehículos estar conectados a la infraestructura fija de modo que cualquier vehículo pueda conectarse con cualquier *host* de internet.

3.3.1- Funcionamiento de una OBU (*On-Board Unit*)

Las OBUs pueden verse como *routers* IPv6 móviles (que implementan la RFC 3963 [18], es decir, implementan NEMO, *Network MObility*) con al menos 2 interfaces de red, una para la red interna del vehículo a la que se conectan las unidades de aplicación (AU), denominada interfaz "*ingress*" y otra, denominada interfaz "*egress*" que utiliza para comunicarse con otras OBUs o con las RSUs. Por ello, nos centraremos en las interfaces "*egress*" de las OBUs, ya que son las únicas que pertenecen al dominio de la red VANET, y por lo tanto, las que implementan el nivel de encaminamiento geográfico.

Para la integración de IPv6, las OBUs deben comportarse como un *host* desde el punto de vista de IPv6 con algunas peculiaridades por tener al nivel de encaminamiento geográfico como "nivel de enlace":

- Capacidad de comunicación con otros nodos IPv6.

- Capacidad de autoconfiguración de una dirección IPv6 mediante el mecanismo de IPv6 *Stateless Address Autoconfiguration*(RFC 4862 [19]): El identificador utilizado para formar los últimos 64 bits de la dirección IPv6 resultante es el identificador del nivel de encaminamiento geográfico, que como habíamos visto, tiene una longitud de 64 bits.

- El identificador del nivel de encaminamiento geográfico se supone que es único en toda la red VANET, por ello, es necesario deshabilitar el mecanismo de detección de direcciones duplicadas de IPv6 (DAD). Además, para mantener la condición de unicidad de las direcciones IPv6, no se soporta la configuración manual de las interfaces "*egress*" de las OBUs.

- Dado que es inmediato pasar de direcciones IPv6 a direcciones del nivel de encaminamiento geográfico, se puede realizar la resolución de direcciones sin la necesidad de mensajes de *Neighbour Solicitation* y *Neighbour Advertisement*. Por lo tanto, no está permitido el envío de estos mensajes. Así se consigue evitar la sobrecarga que generaría en la red VANET el envío de estos paquetes *broadcast*. La resolución de direcciones se realiza de forma que cuando un nodo desea mandar un paquete IPv6 a un destino con una dirección IPv6 de su mismo prefijo, se obtiene el identificador de nivel de encaminamiento geográfico del destino (que son los últimos 64 bits de la dirección IPv6), y se le hace llegar directamente sin necesidad de pasar por la RSU (que tiene el papel de *router* de acceso). Si la dirección IPv6 del destino tiene un prefijo distinto al del nodo, el paquete se le envía a la RSU (*router* de acceso) para que se encargue de encaminar el paquete apropiadamente.

3.3.2- Funcionamiento de una RSU (*Road Side Unit*)

Las RSUs se comportarán a nivel de IPv6 como los *routers* de acceso al que las OBUs se vinculan. Una RSU tendrá al menos dos interfaces de red, una conectada a la infraestructura fija para poder proveer acceso a internet, y otra interfaz en la que opera el nivel de encaminamiento geográfico para las comunicaciones con la VANET.

Al comportarse como un *router* de acceso, la RSU enviará periódicamente paquetes *Router Advertisement* con los prefijos de las direcciones IPv6 asociados al área geográfica que cubren. De esta manera las OBUs podrán configurar de forma automática una dirección IPv6 tomando el prefijo difundido por la RSU encargada de la zona geográfica donde se encuentran y su identificador del nivel de encaminamiento geográfico (nivel que se encuentra justo debajo de IPv6).

Para la integración de IPv6, las RSUs deben comportarse como un *router* desde el punto de vista de IPv6 con algunas peculiaridades por tener al nivel de encaminamiento geográfico como "nivel de enlace":

- Para generar la dirección IPv6 *link-local* del interfaz de red que pertenece a la red VANET se utiliza el identificador del nivel de encaminamiento geográfico.

- No es necesario el envío de paquetes *Neighbour Solicitation* y *Neighbour Advertisement* para realizar la resolución de direcciones debido a que es inmediato pasar de direcciones IPv6 a direcciones del nivel de encaminamiento geográfico.

- La dirección IPv6 global del interfaz de red que pertenece a la red VANET se debe configurar manualmente, pero siempre teniendo en cuenta que como identificador de interfaz (los últimos 64 bits de la dirección IPv6) se toma el identificador de nivel de encaminamiento geográfico.

- Se deshabilita el mecanismo de detección de direcciones duplicadas (DAD) por la condición de unicidad de los identificadores de nivel de encaminamiento geográfico dentro de la red VANET.

3.3.3- Concepto de *link* y el caso particular de la dirección *all-nodes* (FF02::1) según GeoSAC

En este proyecto, para la integración de IPv6 sobre el protocolo de encaminamiento geográfico, se ha utilizado el concepto de *link* de GeoSAC [16]. Para IPv6, el nivel de encaminamiento geográfico es como su nivel de enlace. De esta forma, el nivel de encaminamiento geográfico recibe de IPv6 el datagrama IPv6 que se quiere enviar y la dirección o identificador destino de nivel de encaminamiento geográfico. Así, el paquete se encapsula en la cabecera del protocolo de encaminamiento geográfico y se encamina en función de la posición del destino. En el siguiente salto IPv6, el nivel de encaminamiento geográfico elimina su cabecera y transfiere el paquete al nivel IPv6. Hay que notar que de esta forma, dos vecinos IPv6 pueden estar separados por más de un salto de nivel de encaminamiento geográfico, pero para IPv6 esto es transparente y es como si estuvieran en el mismo enlace. En la figura 16 se puede ver gráficamente este concepto.

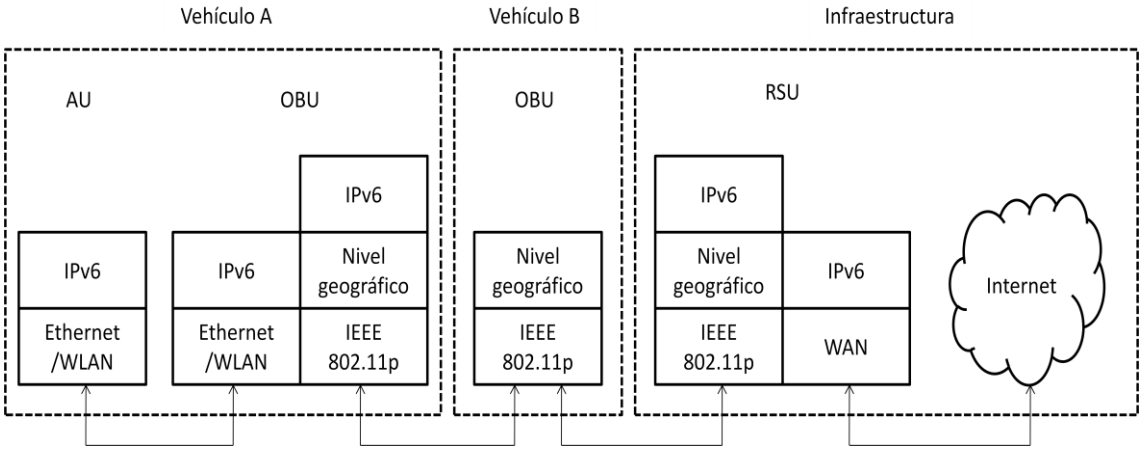

Figura 16 : Integración de IPv6 sobre el nivel de encaminamiento geográfico

En la figura 16, el nivel IPv6 de la OBU del vehículo A y la RSU son vecinos IPv6 y es como si estuvieran en el mismo enlace aunque en realidad, ha sido necesario que el nivel de encaminamiento geográfico del vehículo B tenga que retransmitir los paquetes y establezca un puente multisalto entre ellos.

De esta forma y debido a las características de las redes VANET, el concepto de *link* tal y como lo entendemos a nivel IPv6 deja de existir. Por el contrario, se puede definir un nuevo concepto de *link* que haría referencia a aquellas zonas geográficas donde se difunden los paquetes IPv6 dirigidos a la dirección *all-nodes* FF02::1, como por ejemplo, los paquetes *Router Advertisement* difundidos por las RSUs. Así, el *link* se define como una zona geográfica delimitada donde el nivel de encaminamiento geográfico hace posible que los paquetes lleguen a todos los nodos o vehículos que se encuentren dentro de ella.

Tal y como se ha mencionado anteriormente, las RSUs tienen el cometido de servir de *router* de acceso a las OBUs que se encuentran dentro de su área geográfica de influencia. De esta manera, utilizando el nuevo concepto de *link*, podemos decir que la RSU y las OBUs que están dentro de su zona geográfica de influencia se encuentran en el mismo *link*. Cabe señalar que las diferentes zonas geográficas que cubren las RSUs no se solapan entre sí y de este modo, se asignan diferentes prefijos IPv6 a áreas diferentes.

Como hemos visto, una OBU podía configurar de forma automática su dirección IPv6 tomando el prefijo difundido por la RSU encargada de la zona geográfica donde se encuentra y su identificador del nivel de encaminamiento geográfico (nivel que se encuentra justo debajo de IPv6). Si la OBU desea enviar un paquete *unicast* a una dirección IPv6 global determinada, la resolución de direcciones es inmediata debido al hecho de que los últimos 64 bits de la dirección IPv6 son el identificador del nivel de encaminamiento geográfico. De esta forma, si la dirección IPv6 del destino es de su mismo prefijo, se le hace llegar el paquete directamente sin necesidad de pasar por la RSU, ya que el nodo emisor y receptor se encuentran en el mismo "*link*". Por el contrario, si la dirección IPv6 del destino pertenece a un prefijo distinto (el nodo destino no se encuentra bajo el área cubierta por la RSU), el siguiente salto IPv6 es la RSU (*router* de acceso). Una vez que el paquete *geounicast* se entrega a la RSU, ésta se encarga de encaminar el paquete apropiadamente hacia el destino por la infraestructura fija de internet.

Cabe preguntarse cómo, el nivel de encaminamiento geográfico de la OBU emisora, puede obtener la posición del destino para poder encaminar el paquete. La respuesta es que utiliza un servicio de localización mediante el cual, a partir del identificador del nivel de encaminamiento geográfico del destino, se puede obtener su posición geográfica. Los mecanismos del servicio de localización no se encuentran dentro de nuestro ámbito de estudio y en la implementación realizada se ha supuesto que todos los nodos conocían la situación geográfica del resto de nodos que forman la red VANET.

Por otro lado, las direcciones IPv6 *multicast* del tipo FF0X::X deben tener un trato especial y para la resolución de direcciones se utiliza un identificador especial de nivel de encaminamiento geográfico.

En la implementación se ha tratado únicamente el caso particular de la dirección todos los nodos del enlace (*all-nodes*, FF02::1), que es la que utilizan las RSUs para difundir los paquetes *Router Advertisement* dentro de su zona geográfica y que, de esta manera, las OBUs puedan configurar de manera automática su dirección IPv6.

Las RSUs difunden los paquetes *Router Advertisement* dentro de su área geográfica de influencia, que se encuentra bien definida y configurada. Por lo tanto, cuando una RSU envía un *Router Advertisement* a la dirección *all_nodes* (FF02::1), en la resolución de direcciones, ésta se traduce a un identificador especial del nivel de encaminamiento geográfico destinado a esta dirección IPv6 *multicast*. Este identificador especial, a su vez, se traduce al área geográfica destino que cubre la RSU y que ha sido configurada manualmente de antemano. Así, el paquete *Router Advertisement* es encapsulado en un paquete *geobroadcast* del nivel de encaminamiento geográfico cuya área geográfica destino es la que cubre la RSU. El nivel de encaminamiento geográfico se encarga de hacer llegar el paquete *Router Advertisement* a todos los nodos dentro del área geográfica destino.

El caso de las OBUs es diferente puesto que pueden cambiar de zona geográfica y por lo tanto no pueden tener configurado de antemano un área geográfica destino para las direcciones IPv6 *multicast* (que recordemos, se traducen a identificadores especiales a nivel de encaminamiento geográfico). Es por esto que cuando una OBU llega a una nueva zona geográfica servida por una nueva RSU, además de configurar una nueva dirección IPv6 ante la llegada del *Router Advertisement*, aprende del paquete *geobroadcast* el área geográfica vinculada al identificador especial de la dirección *all-nodes* (FF02::1). De esta manera, si la OBU se encuentra en movimiento, según vaya cambiando de RSUs, irá configurando nuevas direcciones IPv6 con diferentes prefijos e irá aprendiendo el área geográfica (o "*link*") vinculada a la dirección *all-nodes*.

Capítulo 4

Herramienta utilizada en el desarrollo del proyecto: *Click Modular Router*

Para el desarrollo de la implementación del protocolo de encaminamiento geográfico y su integración con IPv6 se necesitaba una herramienta que permitiera poder ejecutar código en el *kernel* de Linux de manera sencilla. La idea era poder capturar y procesar los paquetes procedentes de las interfaces de red antes de que fueran procesados por Linux. Del mismo modo, se necesitaba que los paquetes procedentes de la pila de red de Linux pudieran ser capturados y procesados antes de que llegaran a las interfaces de red del ordenador. *Click Modular Router* [23] cumple con estas condiciones y es por ello que ha sido la herramienta utilizada para el desarrollo del proyecto.

4.1- Introducción a *Click Modular Router*

Click Modular Router es un software para construir *routers* que sean flexibles y configurables. En *Click*, un *router* está formado por el ensamblaje de diferentes módulos que procesan los paquetes. A estos módulos se les llama elementos.

La idea es que cada elemento, de forma individual, realice una tarea simple, como puede ser la clasificación de paquetes en función de alguna característica, espera en cola de paquetes, planificación de paquetes para, por ejemplo, temas de calidad de servicio, interacción con las interfaces de red del equipo, etc.

La forma de ensamblar los diferentes elementos es mediante un archivo de configuración que se traduce a un grafo dirigido en el que los vértices son los diferentes elementos que forman el *router* y las aristas indican las conexiones entre ellos. De esta forma, las aristas señalan la dirección en la que fluyen los paquetes de elemento en elemento. Para escribir este archivo de configuración se utiliza un lenguaje que se basa en declaraciones de elementos y conexiones entre ellos. Además, el lenguaje soporta mecanismos de abstracción de fragmentos de configuración para hacer al usuario la tarea de la descripción del *router* más fácil.

4.2- Elementos de *Click Modular Router*

El elemento es la abstracción visible al usuario más importante. Cada propiedad de la configuración de un *router* viene determinada por las características de los elementos que la

forman y por su ordenación dentro de la misma. Dentro de un *router* cada elemento es un objeto C++ y las conexiones entre ellos se consiguen mediante llamadas a funciones virtuales que toman como parámetro un puntero a la estructura del paquete que se quiera procesar.

4.2.1- Propiedades de los elementos de *Click Modular Router*

Los elementos de *Click Modular Router* tienen cinco propiedades importantes:

- Clase de elemento: la clase del elemento determina el comportamiento del mismo, al igual que les ocurre a los objetos en los lenguajes de programación orientados a objetos. Por ejemplo, indica cuántos puertos tiene, qué manejadores soporta, cómo procesa los paquetes, etc.

- Puertos: cada elemento puede tener cualquier número de puertos de entrada y de salida. Los puertos son los puntos finales de las conexiones entre elementos y cada uno de ellos tiene un rol diferente. Por poner un ejemplo, muchos elementos utilizan su primer puerto de salida para los paquetes que son procesados correctamente y su segundo puerto de salida para aquellos paquetes con errores en su tratamiento.

- Cadena de configuración: algunos elementos soportan argumentos adicionales para configurar el elemento en su inicialización. La forma de hacer llegar al elemento estos parámetros es mediante la cadena de configuración. Los parámetros que se introducen en la cadena de configuración sirven para determinar el comportamiento del elemento, tal y como hacen los parámetros pasados a los constructores de objetos en los lenguajes de programación orientados a objetos.

- Métodos de interfaz: cada elemento exporta métodos que otros elementos pueden llamar para interactuar entre ellos. Todos los elementos están obligados a exportar una serie de métodos que se utilizan, entre otras cosas, para transferir los paquetes entre unos elementos y otros. Además, un elemento puede exportar métodos que no sean obligatorios, como por ejemplo, el elemento *Queue*, que implementa una cola FIFO y que exporta un método que permite conocer su longitud actual.

- Manejadores: los elementos de *Click Modular Router* añaden la posibilidad de poder interactuar con ellos una vez que han sido inicializados y el *router* se encuentra en funcionamiento. Esto se puede hacer gracias a los manejadores. Con los manejadores, se puede por ejemplo, cambiar la configuración de un elemento o conocer sus valores. Por ello, existen manejadores de lectura y de escritura, que sirven para obtener información o cambiar la configuración de los elementos del *router* respectivamente. Todo esto se realiza sin necesidad de parar la ejecución del *router* para volver a inicializarlo.

Click Modular Router ofrece una gran variedad de elementos ya implementados que se encuentran clasificados por función en diferentes grupos como pueden ser *Ethernet*, ARP, IPv4, IPv6, etc.

4.2.2- Paquetes en *Click Modular Router*

Los paquetes en *Click Modular Router* son punteros que señalan a una estructura de cabecera que mantiene a su vez punteros a los datos actuales del paquete, es decir, el contenido del paquete, y a una serie de anotaciones que tienen los paquetes.

Anotaciones de los paquetes

Las anotaciones de los paquetes son información adicional adjunta al paquete y que no se encuentra dentro de los datos del paquete. Las anotaciones se utilizan para procesar de una manera más fácil los paquetes. Existen muchos tipos de anotaciones como por ejemplo, la anotación *paint*, que se utiliza para marcar a los paquetes con un cierto número por algún motivo, como puede ser el número del puerto de la interfaz de red de entrada. Otros tipos de anotaciones son por ejemplo las anotaciones de marcas de tiempo, que puede servir para anotar el tiempo de llegada de los paquetes. También, anotación de la dirección IP de destino, etc. Existen multitud de tipos de anotaciones de paquetes.

Procesamiento de los paquetes

Click Modular Router proporciona dos tipos de conexiones entre elementos. Las conexiones pueden ser *push* o *pull*. En una conexión *push*, el elemento *upstream* le transfiere el paquete al elemento que se encuentra *downstream* (siguiente elemento). En una conexión *pull*, el elemento *downstream* pregunta al elemento *upstream* (elemento anterior) por la llegada de paquetes antes de que éste se los transfiera. En la figura 17 [24], se puede observar esquemáticamente el comportamiento de ambos tipos de conexión.

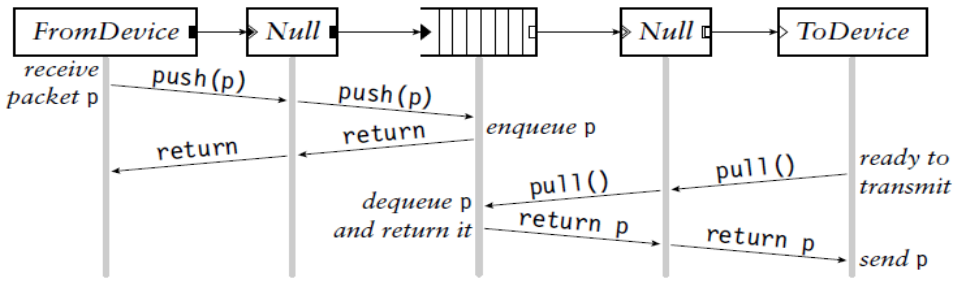

Figura 17 : Funcionamiento de las conexiones *push* y *pull*

Por otro lado, además de que los elementos tengan puertos *push* y *pull*, un puerto puede ser declarado como *agnostic*. Que un puerto sea *agnostic* quiere decir que su comportamiento puede ser como el de un puerto *push* o un puerto *pull* en función de los elementos a los que se encuentre conectado, es decir, en función del contexto del *router*. De esta forma, los puertos que tienen comportamiento *push* sólo pueden ser conectados a puertos con comportamiento *push* y los puertos con comportamiento *pull* sólo pueden conectarse a otros puertos *pull*. Si en el archivo de configuración se conecta un puerto *push* con otro *pull*, se producirá un error y el *router* no podrá ser inicializado.

A los elementos que no intervienen en el procesamiento de paquetes y que simplemente almacenan y mantienen alguna información o dato, se les llama elementos de información.

4.2.3- Planificación de elementos

Un elemento además de ser activado ante peticiones de *push* o de *pull*, puede ser colocado en la cola de planificación del *router Click* para que realice una operación en un instante de tiempo determinado.

4.3- Configuraciones en *Click Modular Router*

Los elementos utilizados en los archivos de configuración de *Click Modular Router* son instanciados y configurados con los parámetros apropiados. Estos parámetros se le pasan al elemento mediante la cadena de configuración en la que se indica una etiqueta que hace referencia a un determinado parámetro y su valor.

En el archivo de configuración, además de indicar mediante declaraciones los elementos que forman el *router* y sus cadenas de configuración, se establecen las conexiones entre ellos. De esta forma, el archivo de configuración se puede ver como un grafo dirigido donde los vértices son los diferentes elementos que forman el *router* y las aristas indican las conexiones entre ellos. De esta manera, las aristas señalan el camino que siguen los paquetes de elemento en elemento.

4.4- Entornos de ejecución

Una configuración de *Click Modular Router* puede correr como un módulo del *kernel* de Linux, como una aplicación a nivel de usuario o en el simulador *ns-2*. Estos entornos de configuración tienen una serie de ventajas e inconvenientes que hay que evaluar en cada caso concreto de uso para determinar cuál es la mejor opción. Los archivos de configuración varían en función del entorno de ejecución que queramos utilizar en el sentido de que, hay elementos que soportan ser utilizados en cualquier entorno de ejecución, pero hay otros que sólo están destinados a ser utilizados en el entorno a nivel de usuario o como módulo del *kernel*. Las principales diferencias se encuentran en los elementos que actúan como fuentes o sumideros de paquetes y los que interactúan con los dispositivos de red del ordenador.

4.4.1- Nivel del *kernel*

Para que una configuración de *Click Modular Router* pueda ser ejecutada a nivel del *kernel* de Linux, el *kernel* debe ser parcheado y compilado para permitir a los elementos de *Click*, codificados en lenguaje C++, ser compilados y ejecutados dentro de un módulo del *kernel*. De este modo, el módulo del *kernel* resultante puede ser instalado con la instrucción de Linux “insmod” (que sirve para instalar módulos en el *kernel*) o más fácilmente con el comando

"click-install" que, además de instalar el módulo en el *kernel*, monta el sistema de ficheros del *router Click*.

El sistema de ficheros del *router Click* es similar al sistema de ficheros */proc* de Linux y sirve para comunicarse con el *router* que se está ejecutando como un módulo del *kernel*. En este sistema de ficheros es dónde se encuentran los ficheros manejadores que sirven para cambiar la configuración del *router* mientras que éste está en ejecución sin necesidad de volver a inicializarlo.

Para aquellos elementos que vayan a ejecutarse a nivel del *kernel*, hay que tener en cuenta todas las limitaciones propias de trabajar en este entorno, como puede ser, que no se permite operar con números en coma flotante o la utilización de librerías de nivel de usuario.

La gran ventaja de trabajar a nivel del *kernel* es que la pila de red de Linux es sustituida por el módulo *Click*. De esta forma, los paquetes entrantes por las interfaces de red son procesados en primer lugar por *Click Modular Router* y Linux sólo podrá procesar aquellos paquetes que nuestro módulo del *kernel* le envíe. De este modo, tenemos la posibilidad de que haya paquetes que sólo sean vistos por *Click Modular Router* y Linux no sepa de su existencia. De la misma manera, todos los paquetes procedentes de la pila de red de Linux pueden ser capturados por *Click Modular Router* antes de que lleguen a las interfaces de red del ordenador. Esta forma de actuar es única para el entorno a nivel del *kernel* y no aplica al entorno de nivel de usuario. Recordemos que éstos eran los requisitos que pedíamos a la herramienta utilizada para desarrollar la implementación y que por ello se ha empleado *Click Modular Router*.

Cabe señalar que únicamente es posible utilizar *Click Modular Router* a nivel del *kernel* para ciertas versiones del mismo. Esto se debe a que únicamente existen parches del *kernel* para las versiones 2.6.24.7, 2.6.19.2, 2.6.16.13, 2.4.32, 2.4.28, 2.4.27, 2.4.26, 2.4.26-Debian, 2.4.21, 2.4.20, 2.4.19 y 2.4.18.

4.4.2- Nivel de usuario

Si lo que se quiere es ejecutar *Click Modular Router* a nivel de usuario, se puede compilar y ejecutar el código de los elementos de forma sencilla sin necesidad de parchear el *kernel*. La diferencia más importante respecto a la ejecución de Click *Modular Router* como módulo del *kernel* es que, a nivel de usuario, *Click* obtiene los paquetes una vez que ya han sido procesados por la pila de red de Linux. De esta forma, otros programas como por ejemplo *sniffers* de paquetes, pueden ver los paquetes aunque estos sean descartados en el *router Click*. Por otro lado, la interacción con los manejadores se puede llevar a cabo a través de un elemento llamado *ControlSocket* que establece una comunicación con el manejador a través de un *socket*.

4.4.3- Entorno de simulador

Para poder ejecutar *Click Modular Router* sobre el simulador *ns-2*, además de la necesidad de tener instalado el simulador, se debe aplicar el parche *nsclick*. De esta manera, los elementos compilados para la simulación pueden ser utilizados en un archivo de configuración de *Click* y ser cargados en el *script* .tcl del simulador.

4.4.4- SMP *Click*

Normalmente *Click Modular Router* se ejecuta en sistemas de un único procesador y utilizando un entorno de un único hilo de ejecución. Sin embargo, es posible utilizar SMP *Click* para poder ejecutar *Click Modular Router* con múltiples hilos ejecutándose en paralelo sobre múltiples procesadores.

4.5- Ejemplo de configuración de *Click Modular Router*

Como ejemplo de una configuración *Click Modular Router* representada como un grafo se muestra la figura 18 [24]. En esta figura se representa un *router* IP para paquetes *unicast*. En este ejemplo, se han tomado únicamente dos interfaces de red aunque el *router* se puede extender para trabajar con más interfaces:

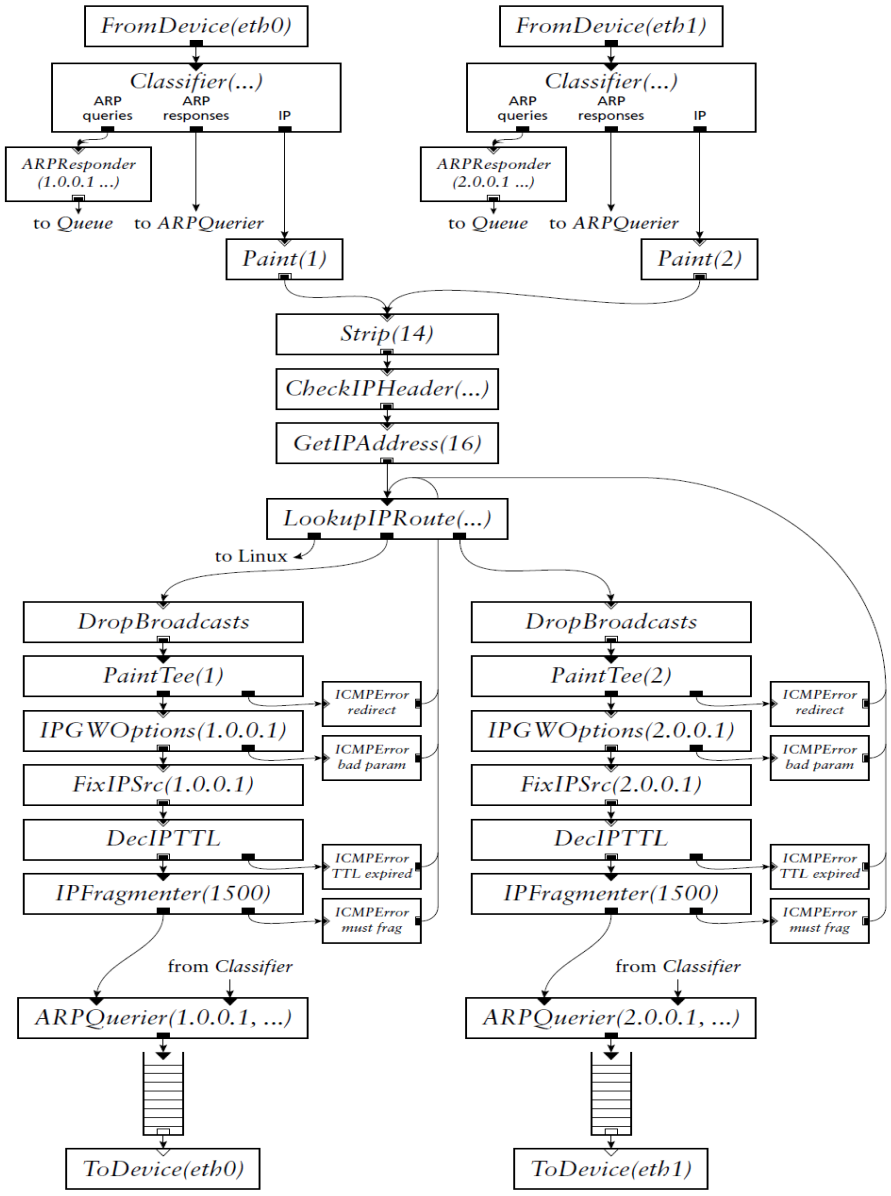

Figura 18 : Ejemplo de la configuración de un *router* IP

Capítulo 5

Implementación del nivel de encaminamiento geográfico y su integración con IPv6

5.1- División en módulos

En la figura 19, se muestra la división en módulos de la implementación realizada.

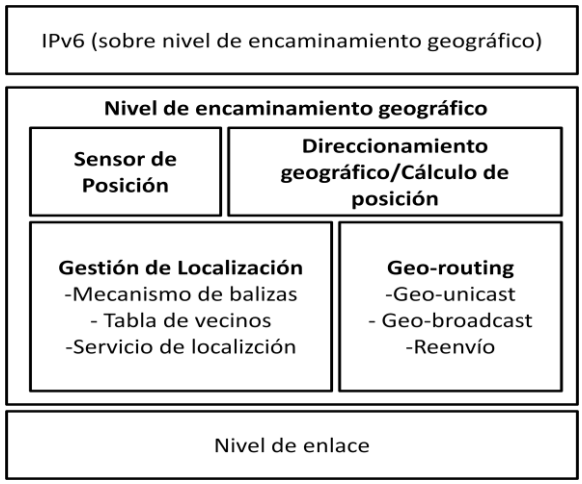

Figura 19 : División en módulos de la implementación

A continuación se explica brevemente la función de cada módulo:

- IPV6 (sobre nivel de encaminamiento geográfico): Se trata del nivel IPv6 con las modificaciones mencionadas en el apartado 3.3 para que pueda ser integrado con el nivel de encaminamiento geográfico. Cabe señalar que en la implementación, se consigue que las RSUs envíen los paquetes *Router Advertisement*, mediante la utilización del demonio RADVD (*Router ADVertisement Daemon*) [27]. Su instalación y configuración se explican en el apéndice B.

- Sensor de posición: este módulo se encarga de obtener la posición actual del vehículo. En un caso real, este módulo utilizaría un sistema GPS que le permitiera obtener todos los datos de posición y precisión necesarios para rellenar los campos de la sección *Position Vector* la cabecera común del nivel de encaminamiento geográfico. En la implementación realizada, éste módulo calcula la posición actual a partir de una posición y velocidad iniciales que se leen de un fichero. Como se supone que la

velocidad del vehículo se mantiene constante, se puede obtener la posición actual a partir del tiempo transcurrido desde el comienzo de la ejecución. Por ello, los campos de precisión de la sección *Position Vector* la cabecera común del nivel de encaminamiento geográfico carecen de importancia, aunque sí que se permite otorgarles un valor (obtenidos también de la lectura del fichero), para posibles ampliaciones de funcionamiento futuras.

- Direccionamiento geográfico/ Cálculo de posición: este módulo se encarga de calcular a partir de la posición actual del vehículo, distancias a otros puntos geográficos, o determinar si el vehículo se encuentra dentro de una zona geográfica determinada.

  Los cálculos de distancia entre dos puntos A y B se realiza de la siguiente forma:

$$\Delta Lat = LatA - LatB$$
$$\Delta Long = LongA - LongB$$
$$a = sin^2(\Delta Lat/2) + \cos(LatA) \cdot \cos(LatB) \cdot sin^2(\Delta Long/2)$$
$$c = 2 \cdot atan2(\sqrt{a}, \sqrt{1-a})$$
$$distancia = R \cdot c$$

  Donde R es el radio de la Tierra (6371000m) y atan2(y,x) devuelve el arcotangente de y/x.
  La distancia se obtiene en metros y las latitudes y longitudes están expresadas en radianes.

  De esta forma, cuando se recibe un paquete *geobroadcast*, también se puede calcular si el nodo se encuentra dentro de la región destino.

- Gestión de localización: se encarga de la gestión del algoritmo de balizas y de mantener en una tabla la lista de los nodos vecinos (aquellos que se encuentran a un salto, es decir, dentro del radio de cobertura) con sus datos de posición geográfica. Por otro lado se encarga del servicio de localización, es decir, el servicio que nos proporciona la posición geográfica del destino a partir de su identificador del nivel de encaminamiento geográfico. Los mecanismos del servicio de localización no se encuentran dentro de nuestro ámbito de estudio y en la implementación realizada se ha supuesto que todos los nodos conocían la situación geográfica del resto de nodos que forman la red VANET. Es decir, cada nodo guarda en una tabla la posición geográfica de todos los nodos de la red para poder obtener la posición geográfica del destino cuando desee enviar un paquete *geounicast*.

- *Geo-routing*: es el módulo principal que implementa el protocolo de encaminamiento geográfico y que hace uso del resto de módulos de su nivel.

- Nivel de enlace: es el nivel de enlace que en un caso real sería alguna tecnología radio *wireless* (IEEE 802.11p, IEEE 802.11 a/b/g/n), pero que en nuestro caso será un nivel *Ethernet*. El hecho de utilizar una red cableada en lugar de una tecnología *wireless*, provoca que tengamos que proveer de un mecanismo que simule un radio de cobertura de los nodos. De esta manera, todos los paquetes que provengan de algún nodo que se

encuentre más lejos de una determinada distancia (que podrá ser modificada para cambiar el radio de cobertura del nodo) serán descartados antes de procesarlos.

5.2- División en elementos de *Click Modular Router*

Para realizar la implementación se ha utilizado la herramienta *Click Modular Router*. De entre los entornos de ejecución que se permiten se ha elegido la ejecución de *Click* como un módulo del *kernel*. La razón por la que se ha elegido este entorno de ejecución es que trabajando a nivel del *kernel*, la pila de red de Linux es sustituida por el módulo *Click* de forma que, los paquetes entrantes por las interfaces de red son procesados en primer lugar por *Click Modular Router* y Linux sólo podrá procesar aquellos paquetes que nuestro módulo del *kernel* le envíe. De esta manera, tenemos la posibilidad de que haya paquetes que sólo sean vistos por *Click Modular Router* y Linux no sepa de su existencia. Del mismo modo, todos los paquetes procedentes de la pila de red de Linux pueden ser capturados por *Click Modular Router* antes de que lleguen a las interfaces de red del ordenador. Esta forma de actuar es única para el entorno a nivel del *kernel* y no aplica al entorno a nivel de usuario. Por eso, se ha elegido trabajar a este nivel. La gran desventaja es que hay que tener en cuenta todas las limitaciones propias de trabajar en este entorno, como puede ser, que no se permite operar con números en coma flotante o la utilización de librerías de nivel de usuario. En el apartado 5.3 llamado "Solución a las desventajas de la ejecución a nivel del *kernel*" se señala la estrategia seguida para poner solución a esta desventaja.

A continuación se exponen los diferentes elementos de *Click Modular Router* implementados en leguaje C++ que se han utilizado y se explica su función.

5.2.1- Elemento *TimedSource*

Figura 20 : Elemento *TimedSource*

Se trata de un elemento con un único puerto de salida de tipo *push*. Genera paquetes con contenido *DATA* cada *INTERVAL* segundos. Por defecto *INTERVAL* tiene un valor de 500 milisegundos. *DATA* debe tener una longitud mínima de 64 bytes.

Los principales argumentos de la cadena de configuración son:

- *DATA*: de tipo *String*. Son los datos del paquete generado.
- *INTERVAL*: de tipo *Time*. Es el intervalo de tiempo entre la generación de dos paquetes.
- *LIMIT*: de tipo Entero. Límite máximo de paquetes generados. Si es negativo, se generan paquetes sin ningún límite.
- *HEADROOM*: de tipo entero sin signo. Establece la longitud de la cabecera que utiliza *Click Modular Router* en los paquetes.
- *STOP*: de tipo *Boolean*. Si toma valor verdadero se para la generación de paquetes. Por defecto vale falso.

Este elemento se utiliza en la implementación para establecer la tasa de envío de paquetes baliza.

5.2.2- Elemento *SetCommonHeaderBeacon*

Figura 21 : Elemento *SetCommonHeaderBeacon*

Se trata de un elemento con un puerto de entrada y otro de salida de tipo *agnostic*. Se encarga de establecer los campos de la cabecera común del nivel de encaminamiento geográfico de los paquetes baliza y asignarles su valor. El elemento no toma ningún argumento.

5.2.3- Elemento *EtherEncap*

Figura 22 : Elemento *EtherEncap*

Se trata de un elemento con un puerto de entrada y otro de salida de tipo *agnostic*. Encapsula los paquetes que le llegan en una cabecera Ethernet con tipo *TYPE*, dirección origen *SRC* y dirección destino *DST*. Los argumentos de la cadena de configuración son:

- *SRC*: es la dirección MAC origen.
- *DST*: es la dirección MAC destino.
- *TYPE*: es el tipo de protocolo que se encuentra encima de Ethernet. En nuestro caso, el tipo del protocolo de encaminamiento geográfico tiene el valor 0x0707.

Este elemento se utiliza en la implementación para encapsular los paquetes de baliza en una cabecera Ethernet. La dirección origen es la MAC del nodo que envía el paquete, la dirección destino es *broadcast* (FF:FF:FF:FF:FF:FF) para que llegue a todos los nodos dentro del rango de cobertura y el tipo de protocolo, el del nivel de encaminamiento geográfico (0x0707).

El elemento también se utiliza en la implementación para encapsular los paquetes que se dirigen al *kernel* de Linux.

5.2.4- Elemento *Queue*

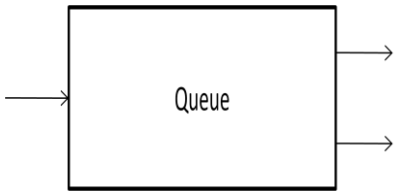

Figura 23 : Elemento *Queue*

Se trata de un elemento con un único puerto de entrada y 1 o 2 de salida. Sigue la estrategia FIFO. Si el número de paquetes en la cola llega a *CAPACITY* se tiran los paquetes entrantes. La cadena de configuración sólo acepta un argumento:

- *CAPACITY*: es el número máximo de paquetes que puede almacenar la cola. Por defecto toma el valor 1000.

Este elemento se utiliza en la implementación para almacenar y planificar los paquetes antes de ser enviados al dispositivo de red.

5.2.5- Elemento *ToDevice*

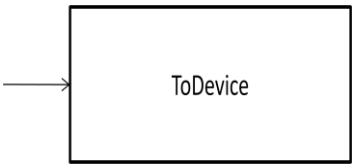

Figura 24 : Elemento *ToDevice*

Tiene un único puerto de entrada de tipo *pull*. Envía los paquetes que recibe a la interfaz de red *DEVNAME*. Los paquetes deben tener cabecera de nivel de enlace y se comprueba que tengan una longitud mínima de 60 bytes.

Los argumentos más importantes que acepta en la cadena de configuración son:

- *QUIET*: de tipo *Boolean*. Si es verdadero, se suprimen los mensajes de habilitación o deshabilitación del interfaz. Por defecto vale falso.
- *ALLOW_NONEXISTENT*: de tipo *Boolean*. Permite asociarse a interfaces de red que no existen. Por defecto vale falso.
- *NO_PAD*: de tipo *Boolean*. Si es verdadero no se fuerza que los paquetes tengan 60 bytes de longitud añadiendo *padding*. Por defecto vale falso.

5.2.6- Elemento *GeoFromHost*

Figura 25 : Elemento *GeoFromHost*

Este elemento tiene un único puerto de salida de tipo *push*. Captura los paquetes que provienen del *kernel* de Linux antes de que lleguen a la interfaz de red. Para ello, crea una interfaz ficticia llamada *DEVNAME*. La peculiaridad de este elemento es que la dirección de la interfaz de red creada es el identificador del nivel de encaminamiento geográfico del nodo. De esta forma se consigue que el nivel IPv6 de Linux consiga autoconfigurar sus direcciones IPv6 tomando como últimos 64 bits el identificador del nivel de encaminamiento geográfico. Cabe señalar, que para conseguir esta funcionalidad, ha sido necesario introducir unos cambios en los archivos *if_arp.h* y *addrconf.c* del *kernel* de Linux.

La interfaz creada puede recibir los paquetes del *kernel* de Linux gracias a que el propio proceso de autoconfiguración de IPv6 modifica la tabla de rutas IPv6 convenientemente.

El argumento que acepta en la cadena de configuración es:

- *DEVNAME*: de tipo *String*. Es el nombre de la interfaz de red ficticia que crea.

Por último, hay que señalar que los paquetes que salen del elemento tienen incorporada la cabecera IPv6.

5.2.7- Elemento *Classifier*

Figura 26 : Elemento *Classifier*

Este elemento clasifica los paquetes según una serie de reglas. Tiene un único puerto de entrada y N puertos de salida, tantos como reglas de clasificación. Todos los puertos son de tipo *push*. Una regla es un patrón que se pasa en la cadena de configuración del elemento y que tiene la sintaxis "desplazamiento/valor" o

"desplazamiento/valor%máscara". Una regla se cumple si los bytes del paquete a partir del desplazamiento coinciden con el valor. El desplazamiento debe ser un número decimal y el valor y la máscara hexadecimales. El dígito "?" está permitido como hexadecimal y significa que ese byte puede tomar cualquier el valor. La regla "-" significa que esa es la salida por defecto. Las reglas se comprueban en orden por lo que aquellas que sean más específicas deben ir primero.

Este elemento se utiliza en dos ocasiones en la implementación: para clasificar los paquetes del nivel de encaminamiento geográfico según su tipo (*beacon*, *geobroadcast* y *geounicast*) y para detectar aquellos paquetes que provienen del *kernel* de Linux que deben ser descartados, como por ejemplo los paquetes *Neighbour Solicitation*, que como habíamos visto, es uno de los requisitos para la integración de IPv6 con el nivel de encaminamiento geográfico.

5.2.8- Elemento *Discard*

Figura 27 : Elemento *Discard*

Este elemento descarta los paquetes que recibe. Tiene un único puerto de entrada de tipo *agnostic*.

Este elemento se utiliza en la implementación cuando es necesario tirar algún paquete.

5.2.9- Elemento *IPV6geoND*

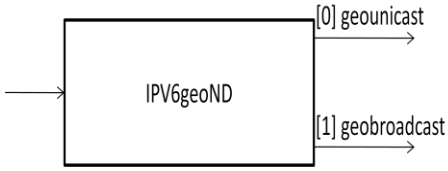

Figura 28 : Elemento *IPV6geoND*

El elemento cuenta con un único puerto de entrada y con dos puertos de salida, todos de tipo *push*. Este elemento es el que recibe los paquetes IPv6 que provienen del *kernel* de Linux. Lo que hace es decidir el tipo de envío necesario comprobando si la dirección IPv6 destino del paquete es de tipo *unicast* (envío *geounicast*) o por el contrario se trata de la dirección IPv6 *multicast all-nodes* (envío *geobroadcast*). Si el envío tiene que ser de tipo *geounicast* se transfiere a la salida 0, en caso contrario, el envío requerido es *geobroadcast* y se transfiere a la salida 1.

En el caso de que el envío sea de tipo *geounicast*, se realiza la resolución de direcciones obteniendo el identificador del nivel de encaminamiento geográfico del siguiente salto IPv6. Este identificador y la correspondiente dirección IPv6 se almacenan en la tabla *IPV6geoNDTable*.

5.2.10- Elemento *IPV6geoNDTable*

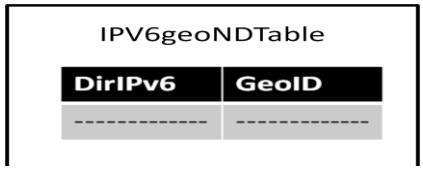

Figura 29 : Elemento *IPV6geoNDTable*

El elemento no cuenta con ningún puerto puesto que no procesa paquetes. Se trata de una tabla para almacenar direcciones IPv6 e identificadores del nivel de encaminamiento geográfico. Esta tabla se utiliza en los envíos de tipo *geounicast* para que el nivel de encaminamiento geográfico obtenga del nivel IPv6 el

datagrama IPv6 y el identificador del nivel de encaminamiento geográfico del siguiente salto IPv6. Los campos de la tabla son *dirección IPv6* y *geoID*.

5.2.11- Elemento *SetDestParam*

Figura 30 : Elemento *SetDestParam*

El elemento cuenta con dos puertos de entrada y con cuatro puertos de salida, todos de tipo *push*. El elemento recibe paquetes para envío *geounicast* por la entrada [0] y paquetes para envío *geobroadcast* por la entrada [1]. Su función es la de establecer los parámetros del destino en función del tipo de envío:

- Para envío *geounicast* se consultan dos tablas: *IPv6GEONDTable* y *LSTable*. De la primera, a partir de la dirección IPv6 destino del paquete de entrada, se obtiene el identificador de nivel de encaminamiento geográfico del siguiente salto IPv6. La segunda tabla actúa como un servicio de localización. A partir del identificador del nivel de encaminamiento geográfico podemos obtener la posición geográfica del destino, para que de esta manera, el paquete *geounicast* pueda ser encaminado. Los paquetes de envío *geounicast* abandonan el elemento por la salida [0].

- Para envío *geobroadcast* se consultan también dos tablas: *IPv6GEONDTable* y *GEOIDTable*. De la primera, a partir de la dirección IPv6 destino del paquete, se obtiene el identificador de nivel de encaminamiento geográfico asociado a ella. En este caso, al ser un envío *geobroadcast*, se tratará de un identificador de nivel de encaminamiento geográfico especial que hace referencia a la dirección IPv6 *all-nodes*(FF02::1). De la segunda tabla podemos obtener los datos de la zona geográfica destino utilizando el identificador especial de nivel de encaminamiento geográfico. Si la zona geográfica destino está descrita como un círculo, el paquete de envío *geobroadcast* abandona el elemento por la salida [1]. Si la zona geográfica destino está descrita como un rectángulo, el paquete de envío *geobroadcast* abandona el elemento por la salida [2].

- Si en las consultas anteriores a las tablas no se encuentra la información buscada, el paquete abandona el módulo por la salida [3] para ser descartado.

5.2.12- Elemento *LSTable*

LSTable

GeoID	Longitud	Latitud	GeoTS
---	---	---	---

Figura 31 : Elemento *LSTable*

El elemento no cuenta con ningún puerto puesto que no procesa paquetes. Se trata de una tabla que simula un servicio de localización, es decir, el servicio que a partir del identificador del nivel de encaminamiento geográfico del destino nos devuelve su posición geográfica. Por ello, la tabla sólo es consultada en envíos de tipo *geounicast*. Esta tabla se configura manualmente en cada nodo y contiene los datos de todos los nodos que forman la red VANET.

Los campos de la tabla son *geoID*, *longitud*, *latitud* y *geoTS* (marca de tiempo). La marca de tiempo sirve para eliminar automáticamente la entrada de la tabla pasado un cierto tiempo de caducidad.

5.2.13- Elemento *GEOIDTable*

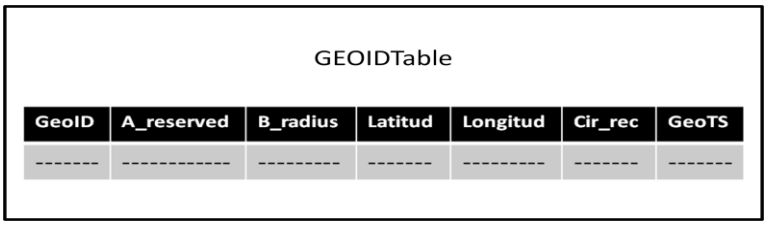

Figura 32 : Elemento *GEOIDTable*

El elemento no cuenta con ningún puerto puesto que no procesa paquetes. Se trata de una tabla para almacenar los datos de las zonas geográficas vinculadas a los identificadores especiales del nivel de encaminamiento geográfico, es decir, aquellos vinculados a direcciones IPv6 *multicast*. En nuestro caso, esta tabla sólo contendrá una entrada puesto que sólo consideramos el caso de la dirección IPv6 *all_nodes* (FF02::1). Esta tabla deberá ser configurada manualmente en aquellos nodos que funcionen como una RSU, ya que es necesario introducir el área geográfica que está bajo su influencia. En el caso de que el nodo sea una OBU, la entrada referente al identificador de la dirección *all_nodes* se irá actualizando cuando se reciba un paquete *Router Advertisement* de la RSU a la que el nodo se tenga que vincular. Los campos de la tabla son *geoID*, *a_reserved*, *b_radius*, *latitud*, *longitud*, *cir_rec* y *geoTS* (marca de tiempo). El campo *a_reseved* se refiere al dato del lado vertical de rectángulo de la zona geográfica. El campo *b_radius* se refiere al dato del lado horizontal de rectángulo o al radio del círculo de la zona geográfica. El campo *cir_rec* es un *flag* que indica si el área geográfica destino está definida como un círculo o como un rectángulo. La marca de tiempo sirve para eliminar automáticamente la entrada de la tabla pasado un cierto tiempo de caducidad.

5.2.14- Elemento *SetSourcePV*

Figura 33 : Elemento *SetSourcePV*

Se trata de un elemento con un puerto de entrada y otro de salida de tipo *agnostic*. Se encarga de establecer los campos de sección *Source node Position Vector* del paquete entrante y asignarles su valor. El elemento no toma ningún argumento.

5.2.15- Elemento *SetCommonHeader*

Figura 34 : Elemento *SetCommonHeader*

Se trata de un elemento con tres puertos de entrada y dos de salida de tipo *push*. Se encarga de establecer los campos de la cabecera común del nivel de encaminamiento geográfico y asignarles su valor. Todo se realiza en función del tipo de paquete entrante. Por la entrada [0] llegan los paquetes de tipo de envío *geounicast*. Por la entrada [1] llegan los paquetes de tipo de envío *geobroadcast* dirigidos a una zona geográfica definida como un círculo. Por la entrada [2]

llegan los paquetes de tipo de envío *geobroadcast* dirigidos a una zona geográfica definida como un rectángulo. Por la salida [0] salen los paquetes de tipo de envío *geounicast*. Por la salida [1] salen los paquetes de tipo de envío *geobroadcast*. El elemento no toma ningún argumento.

5.2.16- Elemento *FromDevice*

Figura 35 : Elemento *FromDevice*

El elemento intercepta todos los paquetes recibidos por el interfaz de red *DEVNAME* (que es uno de los argumentos válido en su cadena de configuración) y los dirige a su único puerto de salida. Los paquetes incluyen la cabecera del nivel de enlace.

5.2.17- Elemento *NextGeoHop*

Figura 36 : Elemento *NextGeoHop*

Se trata del elemento central del encaminamiento geográfico. Cuenta con cinco puertos de entrada y cuatro de salida. Todos ellos son de tipo *push*. El elemento recibe por el puerto de entrada [0] los paquetes baliza, por el puerto de entrada [1] los paquetes *geounicast* procedentes de otros nodos, por el puerto de entrada [2] los paquetes *geobroadcast* procedentes de otros nodos, por el puerto de entrada [3] los paquetes *geounicast* procedentes del *host* y por el puerto de entrada [4] los paquetes *geobroadcast* procedentes del *host*. El elemento procesa los paquetes siguiendo el algoritmo de encaminamiento geográfico (apartado 3.1). Para el procesado de los paquetes el elemento utiliza dos tablas*: NeighbourTable* y *BroadcastPacketTable*. En la primera se almacena información sobre la posición geográfica de los nodos vecinos y en la segunda se almacenan los paquetes *geobroadcast* ya recibidos anteriormente. Tras su procesamiento, los paquetes son enviados a la salida correspondiente. Los paquetes de los cuales el nodo es destinatario, dejan el elemento por la salida [0]. Los paquetes *geounicast* o *geobroadcast* que proceden de otros nodos y que deben ser retransmitidos hacia el destino, dejan el elemento por la salida [1]. Los paquetes *geounicast* o *geobroadcast* procedentes del *host* y que son transmitidos hacia el destino, dejan el elemento por la salida [2]. Los paquetes que no es posible encaminarlos por que no exista un vecino más cercano al destino que el propio nodo o porque sean paquetes *geobroadcast* ya recibidos anteriormente, dejan el elemento por la salida [3] para ser descartados.

5.2.18- Elemento *NeighbourTable*

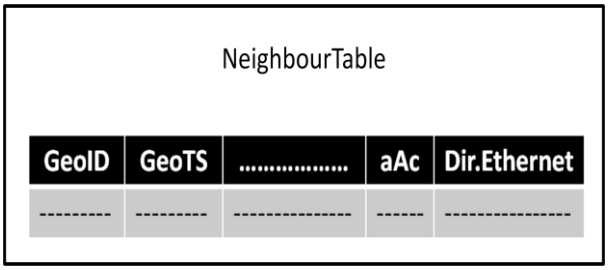

Figura 37 : Elemento *NeighbourTable*

El elemento no cuenta con ningún puerto puesto que no procesa paquetes. Se trata de una tabla para almacenar los datos de la posición geográfica de los nodos vecinos, es decir, aquellos que se encuentra a un salto o dentro del radio de cobertura. Los campos de la tabla son *geoID*, *geoTS* (marca de tiempo), latitud, longitud, *velocidad*, *dirección*, *altitud*, *timeAc*, *posAc*, *sAc*, *hAc*, *aAc* y *dirección Ethernet*. Como se puede observar, son todos los campos de la cabecera común del nivel de encaminamiento geográfico más la dirección *Ethernet*. Es necesario conocer la dirección *Ethernet* de los vecinos para poder retransmitir el paquete de nodo en nodo siguiendo un puente multisalto entre origen y destino. La marca de tiempo sirve para eliminar automáticamente la entrada de la tabla pasado un cierto tiempo de caducidad.

5.2.19- Elemento *BroadcastPacketTable*

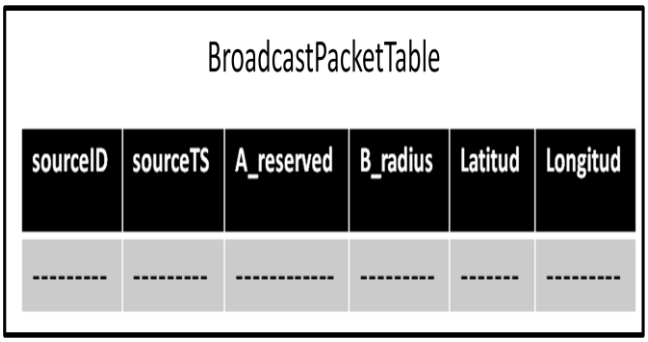

Figura 38 : Elemento *BroadcastPacketTable*

El elemento no cuenta con ningún puerto puesto que no procesa paquetes. Se trata de una tabla que almacena los paquetes *geobroadcast* ya recibidos. Sirve para evitar bucles y para que la inundación simple dentro de la zona geográfica destino cese en algún momento. En realidad no se guarda el paquete *geobroadcast* completo, sino una identificación unívoca del mismo. Por ello, los campos de la tabla son *sourceID*, *sourceTS* (marca de tiempo), *dest_a_reserved*, *dest_b_radius*, *dest_latitude*, *dest_longitude*. Todos estos valores se obtienen de la cabecera *geobroadcast* del paquete. Al igual que en el resto de tablas, las entradas se eliminan automáticamente pasado un cierto tiempo de caducidad.

5.2.20- Elemento *EraseGeoHeader*

Figura 39 : Elemento *EraseGeoHeader*

Se trata de un elemento con un puerto de entrada y otro de salida de tipo *agnostic*. Se encarga de eliminar la cabecera del nivel de encaminamiento geográfico antes de enviar el paquete al nivel IPv6 del *kernel* de Linux. El elemento no toma ningún argumento.

5.2.21- Elemento *ChangeCommonHeader*

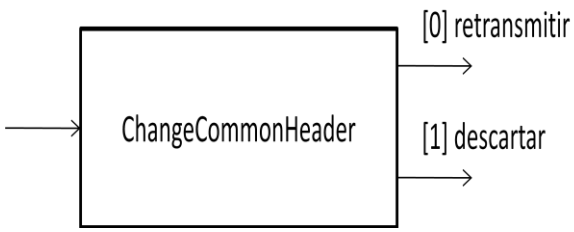

Figura 40 : Elemento *ChangeCommonHeader*

El elemento cuenta con un único puerto de entrada y con dos puertos de salida, todos de tipo *agnostic*. Se encarga de establecer los campos de la cabecera común del nivel de encaminamiento geográfico y asignarles su valor antes de retransmitir el paquete. Una de sus misiones es la de reducir el valor del campo *HopLimit*. Si el paquete puede ser retransmitido deja el elemento por la salida [0]. Si por el contrario, el paquete ha alcanzado su máximo número de saltos, el paquete deja el elemento por la salida [1] para ser descartado. El elemento no toma ningún argumento.

5.2.22- Elemento *SetPacketType*

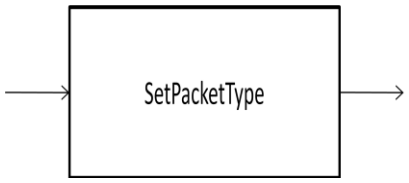

Figura 41 : Elemento SetPacketType

Se trata de un elemento con un puerto de entrada y otro de salida de tipo *agnostic*. Establece la anotación de tipo de paquete del paquete al valor *TYPE* pasado como argumento en la cadena de configuración. Esta anotación es utilizada por el *kernel* de Linux para conocer las características del nivel de enlace. Por ejemplo, ¿el paquete fue enviado al *host* directamente o ha sido recibido por *broadcasting*? *TYPE* puede tomar los valores *HOST, BROADCAST, MULTICAST, OTHERHOST, OUTGOING* o *LOOPBACK.*

Este elemento se utiliza en la implementación para establecer la anotación de tipo de paquete a *HOST*. De otra manera se consigue que el *kernel* de Linux acepte y procese el paquete.

5.2.23- Elemento *MarkIP6Header*

Figura 42 : Elemento *MarkIP6Header*

Se trata de un elemento con un puerto de entrada y otro de salida de tipo *agnostic*. Establece las anotaciones de IPv6 del paquete. Con el argumento *OFFSET*, que se pasa en la cadena de configuración, se le indica en que byte empieza la cabecera IPv6. El valor de *OFFSET* por defecto es 0.

Este elemento se utiliza en la implementación porque es necesario establecer las anotaciones de IPv6 para que el elemento *ToHost* transfiera el paquete al *kernel* de Linux.

5.2.24- Elemento *ToHost*

Figura 43 : Elemento *ToHost*

Este elemento tiene un único puerto de entrada de tipo *push*. Su función es la de mandar los paquetes que recibe al *kernel* de Linux. Si se indica un nombre de interfaz mediante el parámetro *DEVNAME*, los paquetes son entregados al *kernel* de Linux como si provinieran de la interfaz de red con ese nombre. Por defecto, y así es como se utiliza en la implementación, se esperan paquetes encapsulados en una cabecera *Ethernet*, aunque es posible modificar el funcionamiento para que espere paquetes con cabeceras IP. Cabe señalar que como Linux espera paquetes con anotaciones válidas, *ToHost* no transferirá al *kernel* de Linux ningún paquete sin anotaciones.

En la implementación el parámetro *DEVNAME* toma el valor fake0, que es el nombre de la interfaz ficticia creada con dirección igual al identificador del nivel de encaminamiento geográfico del nodo. De esta forma se consigue que el nivel IPv6 de Linux consiga autoconfigurar sus direcciones IPv6 tomando como últimos 64 bits el identificador del nivel de encaminamiento geográfico.

5.2.25- Elemento *MultihopSimulation*

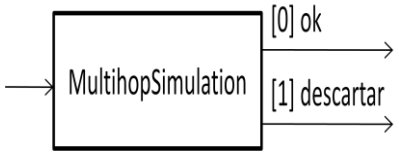

Figura 44 : Elemento *MultihopSimulation*

Este elemento tiene un único puerto de entrada y dos de salida. Todos son de tipo *agnostic*. Como utilizamos una red cableada en lugar de una tecnología *wireless*, es necesario un mecanismo que simule que el nodo tiene un radio de cobertura. Este elemento se encarga de ello. De esta manera, todos los paquetes que provengan de algún nodo que se encuentre más lejos de una determinada distancia (que podrá ser modificada por medio del parámetro RADIUS) serán descartados por la salida [1]. Los paquetes procedentes de nodos dentro del rango de cobertura, salen del elemento por la salida [0].

5.3- Solución a las desventajas de la ejecución a nivel del *kernel*

La gran ventaja de trabajar a nivel del *kernel* es que la pila de red de Linux es sustituida por el módulo *Click*. De esta forma, los paquetes entrantes por las interfaces de red son procesados en primer lugar por *Click Modular Router* y Linux, sólo podrá procesar aquellos paquetes que nuestro módulo del *kernel* le envíe. De esta manera, tenemos la posibilidad de que haya paquetes que sólo sean vistos por *Click Modular Router* y Linux no sepa de su existencia. Del mismo modo, todos los paquetes procedentes de la pila de red de Linux pueden ser capturados por *Click Modular Router* antes de que lleguen a las interfaces de red del ordenador.

Pero trabajar a nivel del *kernel* también presenta una serie de limitaciones, como por ejemplo, que no se permite operar con números en coma flotante o la utilización de librerías de nivel de usuario.

Esto nos afecta a la hora de realizar operaciones matemáticas cuando se calcula la distancia de un nodo a sus vecinos o se determina si nos encontramos dentro de la zona geográfica destino ante la llegada de un paquete *geobroadcast*, etc.

También estamos limitados a la hora de escribir o leer de un fichero por no poder utilizar las librerías del nivel de usuario. Esto supone un gran problema puesto que, los datos de posición, velocidad, altura, dirección y precisiones iniciales se obtienen de un fichero.

Para poner solución a estos problemas se utiliza un proceso auxiliar ejecutándose a nivel de usuario que recibe órdenes del módulo del *kernel*. La forma de poder comunicar un módulo del *kernel* con un proceso que se encuentra en ejecución a nivel de usuario es mediante unos *sockets* especiales llamados *netlinks* [28] que se utilizan de una manera parecida a los *sockets* tradicionales de nivel de usuario.

De esta manera, se ideó un protocolo de intercambio de mensajes entre el módulo del *kernel* y el proceso auxiliar que se ejecuta a nivel de usuario.

5.3.1- Protocolo de comunicación entre el módulo del *kernel* y el proceso auxiliar de nivel de usuario

Se estableció un protocolo de intercambio de mensajes basado en un esquema de petición-respuesta. El módulo del *kernel* envía el mensaje con una petición y el proceso auxiliar le contesta con la respuesta o confirmación tras realizar las operaciones oportunas. El modelo de intercambio de mensajes debe ser bloqueante. Esto se debe a que el módulo del *kernel* no puede continuar con su ejecución hasta que no reciba los datos que necesita. Para establecer que el intercambio de mensajes sea bloqueante se ha utilizado un mecanismo de semáforos.

5.3.1.1- Tipos de peticiones

Las peticiones que el módulo del *kernel* necesita que el proceso auxiliar de entorno de usuario cumpla son:

- Obtener la versión:

  Con este tipo de mensaje, el módulo del *kernel* solicita al proceso auxiliar que le responda con el número de versión que debe asignar al campo *Version* de la cabecera común del nivel de encaminamiento geográfico. Cuando el proceso auxiliar recibe una petición de este tipo, obtiene la información mediante la lectura de un fichero llamado *version.txt*. El fichero se encuentra codificado en formato ASCII y tiene en su primera línea el número de la versión.

- Obtener la prioridad:

  Con este tipo de mensaje, el módulo del *kernel* solicita al proceso auxiliar que le responda con la prioridad de los paquetes que transmite. De esta forma, podrá asignar valor al campo *Prio* de la cabecera común del nivel de encaminamiento geográfico.

Cuando el proceso auxiliar recibe una petición de este tipo, obtiene la información mediante la lectura de un fichero llamado *priority.txt*. El fichero se encuentra codificado en formato ASCII y tiene en su primera línea la prioridad.

- Obtener la potencia de transmisión:

  Con este tipo de mensaje, el módulo del *kernel* solicita al proceso auxiliar que le responda con la potencia de transmisión del paquete que se va a enviar. De esta forma, podrá asignar valor al campo *TxPower* de la cabecera común del nivel de encaminamiento geográfico. Cuando el proceso auxiliar recibe una petición de este tipo, obtiene la información mediante la lectura de un fichero llamado *tx_power.txt*. El fichero se encuentra codificado en formato ASCII y tiene en su primera línea la potencia de transmisión.

- Obtener el identificador de nivel de encaminamiento geográfico del nodo:

  Con este tipo de mensaje, el módulo del *kernel* solicita al proceso auxiliar que le responda con el identificador del nivel de encaminamiento geográfico del nodo. Cuando el proceso auxiliar recibe una petición de este tipo, obtiene la información mediante la lectura de un fichero llamado *sourceID.txt*. El fichero se encuentra codificado en formato ASCII y tiene en su primera línea el identificador del nivel de encaminamiento geográfico del nodo.

- Realizar el cálculo de la distancia entre dos puntos geográficos.

  Con este tipo de mensaje, el módulo del *kernel* solicita al proceso auxiliar que le calcule la distancia existente entre dos puntos geográficos. Los puntos geográficos se encuentran expresados por su longitud y latitud. Cuando el proceso auxiliar recibe una petición de este tipo, realiza el cálculo y le devuelve al módulo del *kernel* el resultado.

- Obtener la posición actual del nodo.

  Con este tipo de mensaje, el módulo del *kernel* solicita al proceso auxiliar que le responda con los datos de posición actual del nodo. Cuando el proceso auxiliar recibe una petición de este tipo, realiza el cálculo de la posición actual. Para ello, se basa en la información que obtiene del fichero de posición inicial. En la primera línea de este fichero podemos encontrar, separados por espacios y en este orden, los siguientes datos:

  Latitud_inicial Longitud_inicial Altitud_inicial Velocidad_inicial Dirección_inicial timeAc posAc sAc hAc aAc

  Estos datos se corresponden con los campos de la sección *Position Vector* de la cabecera común del nivel de encaminamiento geográfico.

  Para calcular la posición actual del nodo, se toma la longitud y latitud iniciales, y a partir del tiempo transcurrido desde el comienzo de la ejecución y la velocidad inicial, se obtienen la longitud y latitud actuales. En este cálculo se supone que la velocidad del nodo se mantiene constante.

- Comprobar si el nodo se encuentra dentro de una zona geográfica determinada

  Con este tipo de mensaje, el módulo del *kernel* solicita al proceso auxiliar que determine si el nodo se encuentra dentro de una región geográfica. El módulo del *kernel* necesita esta comprobación cuando recibe un paquete *geobroadcast*. El proceso auxiliar recibe los datos de la zona geográfica destino del paquete *geobroadcast* y basándose en la posición actual del nodo, realiza la comprobación.

- Clausura del *socket netlink*

  Cuando se desea desinstalar el módulo del *kernel*, es necesario terminar en primer lugar con la ejecución del proceso auxiliar. Cuando esto sucede, el proceso auxiliar envía un mensaje al módulo del *kernel* indicando que va a cerrar el *socket netlink*. De esta manera, el módulo del *kernel* deja de enviar peticiones y puede cerrar el *socket* también. La finalización del proceso auxiliar se realiza mediante la combinación de teclas Control+C. El comando que permite desinstalar el módulo del *kernel* es *click-uninstall*.

5.3.1.2- Formato del mensaje de intercambio

El mensaje con el que el módulo del *kernel* realiza las peticiones y el proceso auxiliar devuelve las respuestas cuenta con los siguientes campos:

- *Type*: indica el tipo de petición del mensaje.

- *Data*: es un *array* que se utiliza para guardar los datos de versión, prioridad, potencia de transmisión e identificador de nivel de encaminamiento geográfico.

- LatitudA*:* este campo se utiliza en tres peticiones. En la petición del cálculo de la distancia entre dos puntos indica la latitud del punto A. En la petición del cálculo de la posición actual del nodo y la de comprobación de pertenencia a una zona geográfica indica la latitud actual del nodo.

- LongitudA*:* este campo se utiliza en tres peticiones. En la petición del cálculo de la distancia entre dos puntos indica la longitud del punto A. En la petición del cálculo de la posición actual del nodo y la de comprobación de pertenencia a una zona geográfica indica la longitud actual del nodo.

- Velocidad: indica la velocidad del nodo en la petición de la posición actual del nodo.

- Dirección: indica la dirección del nodo en la petición de la posición actual del nodo.

- Altitud: indica la altitud del nodo en la petición de la posición actual del nodo.

- *TimeAc*: indica la precisión temporal en la petición de la posición actual del nodo. El campo *TimAc* de la cabecera común de encaminamiento geográfico se establece a este valor.

- *PosAc*: indica la precisión de la posición en la petición de la posición actual del nodo. El campo *PosAc* de la cabecera común de encaminamiento geográfico se establece a este valor.

- *SAc*: indica la precisión de la velocidad en la petición de la posición actual del nodo. El campo *SAc* de la cabecera común de encaminamiento geográfico se establece a este valor.

- *HAc*: indica la precisión de la dirección en la petición de la posición actual del nodo. El campo *HAc* de la cabecera común de encaminamiento geográfico se establece a este valor.

- *AAc*: indica la precisión de la altitud en la petición de la posición actual del nodo. El campo *AAc* de la cabecera común de encaminamiento geográfico se establece a este valor.

- LatitudB*:* este campo se utiliza en dos peticiones. En la petición del cálculo de la distancia entre dos puntos indica la latitud del punto B. En la petición de comprobación de pertenencia a una zona geográfica, indica la latitud del punto central de la región geográfica destino.

- LongitudB*:* este campo se utiliza en dos peticiones. En la petición del cálculo de la distancia entre dos puntos indica la longitud del punto B. En la petición de comprobación de pertenencia a una zona geográfica, indica la longitud del punto central de la región geográfica destino.

- *A_reserved*: este campo se utiliza en la petición de comprobación de pertenencia a una zona geográfica. Cuando el área geográfica considerada es un rectángulo, es la longitud en metros de la arista vertical que define la región destino. Se entiende por arista vertical, aquella que se orienta de norte a sur. Si el área destino del paquete es un círculo, este campo toma el valor 0.

- *B_radius*: este campo se utiliza en la petición de comprobación de pertenencia a una zona geográfica. Cuando el área geográfica considerada es un rectángulo, es la longitud en metros de la arista horizontal que define la región destino. Se entiende por arista horizontal, aquella que se orienta de este a oeste. Si el área destino del paquete es un círculo, este campo indica la longitud en metros del radio del mismo.

- *Cir_rec*: este campo se utiliza en la petición de comprobación de pertenencia a una zona geográfica. Cuando el área geográfica considerada es un rectángulo, toma valor 0. Si por el contrario, se trata de un círculo, este campo toma el valor 1.

- *Inzone*: este campo se utiliza en la petición de comprobación de pertenencia a una zona geográfica. Cuando el nodo se encuentra en al área geográfica considerada, el campo toma el valor 1. En caso contrario, toma el valor 0.

Hay que señalar que tanto en los ficheros como en el mensaje de intercambio, las unidades en que se encuentran expresados los datos son las mismas que las de los campos de la cabecera común de nivel de encaminamiento geográfico.

5.4- Unión de los elementos de *Click Modular Router*

La forma de ensamblar los diferentes elementos que se han descrito en el apartado 5.2 es mediante un archivo de configuración. Éste se puede entender como un grafo dirigido en el que los vértices son elementos y las aristas, indican las conexiones entre ellos. Por lo tanto, la forma de procesar los paquetes depende de cómo se conectan los elementos entre sí y del orden en el que son colocados.

El archivo de configuración de la implementación y el esquema del grafo resultante se pueden consultar en el apéndice D.

Capítulo 6

Evaluación de la implementación

6.1- Introducción al entorno de pruebas

Para evaluar la implementación realizada se optó por la utilización de un software de virtualización.

Un sistema virtual por software es un programa que simula un sistema físico (un ordenador o un dispositivo hardware) con unas características de hardware determinadas. A este sistema virtual se le llama máquina virtual. Cuando se ejecuta una máquina virtual, ésta proporciona un entorno de ejecución similar a un ordenador físico, con CPU, BIOS, tarjeta gráfica, memoria RAM, tarjeta de red, sistema de sonido, conexión USB, disco duro, etc.

Se evaluaron dos software de virtualización distintos para virtualizar el entorno de pruebas, *VMWare Server 2.0* [29] y *Virtual Network User Mode Linux* (VNUML) [30]. La opción elegida fue *VMWare Server 2.0* debido a que las máquinas virtuales soportan una gran variedad de sistemas operativos, incluido Linux con el *kernel* utilizado para ejecutar *Click Modular Router*. Sin embargo, con VNUML, las máquinas virtuales soportan únicamente el sistema operativo Linux con unos sistemas de ficheros y *kernels* precompilados entre los que no se encuentra ningún *kernel* válido para ejecutar *Click Modular Router*. Además, hay que añadir la sencillez con la que se maneja *VMWare Server 2.0* y la gran cantidad de documentación disponible.

Por lo tanto, lo que conseguimos con *VMWare Server 2.0* es ejecutar varios ordenadores o máquinas virtuales (cada una de ellas independiente del resto) dentro de un mismo hardware de manera simultánea. De esta manera, podemos crear un entorno virtual de pruebas en un único ordenador.

Por otro lado, *VMWare Server 2.0* permite crear redes virtuales a las que las máquinas virtuales se pueden conectar. Esto permite emular la interconexión de distintas máquinas virtuales en diferentes topologías de red. Estas redes pueden ser completamente virtuales o pueden estar conectadas a alguna de las interfaces de red del ordenador físico donde se ejecuta el software de virtualización.

6.1.1- Descripción de las máquinas virtuales

Para realizar las pruebas se han utilizado dos tipos de máquinas virtuales, las máquinas virtuales estándar y una máquina virtual especial.

Las máquinas virtuales estándar que son aquellas que ejecutan el código de la implementación de tal manera que se comportan como OBUs o RSUs de la red VANET. Las máquinas virtuales estándar tienen las siguientes características:

- Sistema operativo Linux Ubuntu 8.04
- Número de procesadores: 1
- Versión de *kernel* de Linux 2.6.24.7 con el parche de *Click Modular Router*.
- Memoria RAM: 256 MB.
- Disco duro de 11 GB.
- 2 interfaces de red.

La máquina virtual especial se encarga de generar los ficheros de posición inicial de los nodos de la red VANET y recopilar los resultados de las pruebas realizadas. La máquina virtual especial tiene instalado un servidor NFS (*Network File System*) de manera que las máquinas virtuales estándar pueden acceder a los ficheros de posición inicial y guardar los ficheros de resultados de las pruebas. La máquina virtual especial tiene las siguientes características:

- Sistema operativo Linux Ubuntu 8.04
- Número de procesadores: 1
- Versión de *kernel* de Linux 2.6.24.23
- Memoria RAM: 256 MB.
- Disco duro de 8 GB.
- 1 interfaz de red.

Por otro lado, para la interconexión de las distintas máquinas virtuales, se han definido dos redes virtuales:

- La red 10.0.0.X con topología en bus. A esta red se encuentran conectadas todas las máquinas virtuales estándar por una de sus interfaces. Es la red que se utiliza para interconectar las OBUs y las RSUs, es decir, es la red VANET.

- La red 192.168.0.X con topología en bus. A esta red se encuentran conectadas todas las máquinas virtuales estándar por una de sus interfaces. También se encuentra en esta red la máquina virtual especial. Esta red es la que se utiliza para llevar el control de las máquinas virtuales estándar al realizar las pruebas.

El motivo por el que se han definido dos redes virtuales independientes es para evitar que las pruebas realizadas puedan verse afectadas por las comunicaciones de control existentes entre las máquinas virtuales.

La topología de las redes del entorno de pruebas se muestra en la figura 45.

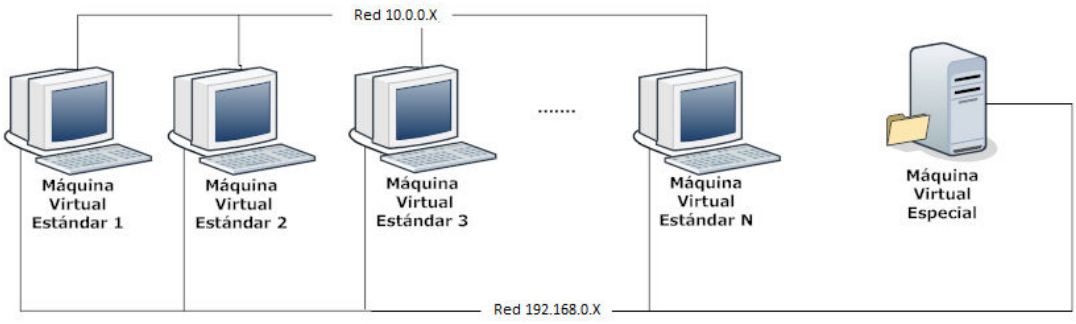

Figura 45: Topología de las redes del entorno de pruebas

6.1.2- Control en la realización de las pruebas

Existen tres problemas a la hora de realizar las pruebas: la sincronización temporal entre diferentes máquinas virtuales para tomar de medidas de tiempo, hacer saber a los nodos de la red VANET cuál es su posición inicial y la recopilación de resultados.

6.1.2.1- Sincronización temporal de las máquinas virtuales

Si se desean realizar medidas de tiempo que vinculen a más de un nodo de la red es necesario sincronizar sus relojes. Para ello, se ha utilizado el protocolo PTP (*Precision Time Protocol*) [31]. PTP es un protocolo parecido a NTP (*Network Time Protocol*) que sirve para sincronizar los relojes de diferentes máquinas entre sí a través de una red, pero que alcanza precisiones mayores. PTP permite alcanzar precisiones de sincronismo por debajo del orden de microsegundos.

En la realización de las pruebas, se instaló el demonio PTPd (*Precision Time Protocol daemon*) [32] en aquellas máquinas que fue necesario sincronizar. La red utilizada para la sincronización es la red virtual de control 192.168.0.X.

La explicación de cómo instalar este programa se encuentra en el apéndice B.

6.1.2.2- Recopilación de resultados y ficheros de posiciones iniciales

La máquina virtual especial se encarga de la recopilación de resultados y ofrecer a las máquinas virtuales estándar sus ficheros de posición inicial en función de la prueba realizada.

La forma en la que las máquinas virtuales estándar consiguen conocer su posición inicial es mediante el montaje del directorio /files/ de la máquina virtual especial. Este directorio contiene los ficheros de posición inicial de todos los nodos de la red VANET para la prueba concreta que se pretende realizar.

La recopilación de resultados se realiza mediante la escritura en un fichero del directorio /files/ de la máquina virtual especial. Cuando una máquina virtual estándar obtiene una medida lo escribe en el fichero oportuno del directorio mencionado.

Para poder hacer todo esto, la máquina virtual especial tiene instalado un servidor NFS que permite a las máquinas virtuales estándar acceder al directorio /files/. De esta manera, las máquinas virtuales estándar pueden montar el directorio /files/ utilizando un cliente NFS.

El protocolo NFS se utiliza sobre la red virtual de control 192.168.0.X.

La explicación de cómo instalar el servidor y el cliente NFS se encuentra en el apéndice B.

6.2- Pruebas realizadas y resultados

A continuación se describen las pruebas realizadas y los resultados obtenidos.

6.2.1- Primer escenario de pruebas: medida del retardo de paquetes *geounicast* desde la RSU a una OBU en función del número de saltos

Con este escenario de pruebas se pretende:

- Validar el funcionamiento de la implementación para el envío de paquetes *geounicast* entre nodos de la red VANET.

- Comprobar el retardo introducido en el reenvío de paquetes desde la RSU a una OBU de la red VANET en función del número de saltos.

El escenario de pruebas se muestra en la figura 46.

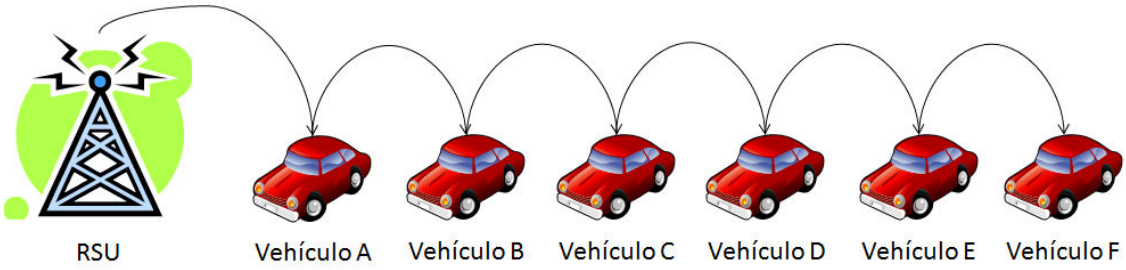

Figura 46 : Primer escenario de pruebas: medida del retardo de paquetes *geounicast* desde la RSU a una OBU en función del número de saltos

En este escenario tenemos una única RSU y una serie de vehículos que se encuentran dentro de su área geográfica de influencia. Ésta tiene forma rectangular y cuenta con unas medidas de 20 metros por 3000 metros.

En el escenario se tienen siete vehículos dispuestos en línea recta, ninguno de los cuales se encuentra en movimiento. Los vehículos se encuentran equiespaciados con una distancia entre

nodos de 200 metros. Esta distancia ha sido escogida intencionadamente para que un nodo sólo tenga comunicación directa con otros dos. De esta manera, un paquete *geounicast* dirigido de la RSU al vehículo F, deberá ser encaminado obligatoriamente por el resto de vehículos hasta llegar al destino, realizando un total de seis saltos. El rango de cobertura de los nodos de la red es de 250 metros.

Para generar las posiciones iniciales de los nodos de la red se ha utilizado un programa en lenguaje C llamado generador_posiciones. Este programa, que se ejecuta en la máquina virtual especial, genera los ficheros de posiciones iniciales siguiendo las premisas del escenario y los almacena en el directorio /files/. De esta manera, las máquinas virtuales estándar pueden saber su posición inicial. La utilización de este programa se describe en el apéndice C.

Hay que señalar que los vehículos en este escenario no se encuentran en movimiento debido a que el servicio de localización consultado para enviar paquetes *geounicast* se encuentra implementado como una tabla estática configurada en cada nodo con los datos de posición de todos los nodos de la red VANET. Por ello, es imposible que los vehículos se puedan mover en pruebas con envíos *geounicast*.

La prueba realizada consiste en enviar paquetes ICMPv6 *Request* desde la RSU hasta el resto de vehículos del escenario, de manera que éstos contesten con paquetes ICMPv6 *Reply*. De esta forma, se puede calcular el tiempo RTT (*Round Trip Time*), es decir, el tiempo de ida y vuelta de los paquetes, en función del número de saltos que deben dar desde el emisor hasta el destino.

Los paquetes ICMPv6 van dirigidos a un único destino, por ello, se tratan de envíos *geounicast.* El número de saltos de un paquete *geounicast* entre la RSU y un vehículo concreto se puede ver en la tabla 3:

	A	**B**	**C**	**D**	**E**	**F**
Número de saltos	1	2	3	4	5	6

Tabla 3 : Número de saltos en función del vehículo destino

Para calcular este tiempo RTT se ha utilizado el comando de Linux *ping6*. Este comando, se encarga de enviar los paquetes ICMPv6 de tipo *Request* y calcular el tiempo RTT medio. Se ha utilizado este comando con sus valores por defecto: intervalo de tiempo entre paquetes ICMPv6 de un segundo y paquetes con 56 bytes de datos (para el nivel de encaminamiento geográfico son 104 bytes de datos: 40 bytes cabecera IPv6 + 8bytes cabecera ICMPv6+56 bytes datos ICMPv6). Se han realizado alrededor de un total de 56400 envíos *geounicast* desde la RSU a cada uno de los vehículos del escenario.

Los resultados obtenidos se muestran en la tabla 4:

	1 salto	**2 saltos**	**3 saltos**	**4 saltos**	**5 saltos**	**6 saltos**
$\overline{\boldsymbol{RTT}}(\boldsymbol{ms})$	1,190	1,581	1,954	2,803	4,307	5,228

Tabla 4 : RTT en función del número de saltos

Para ofrecer una mejor percepción visual, estos resultados se encuentran representados gráficamente en la figura 47.

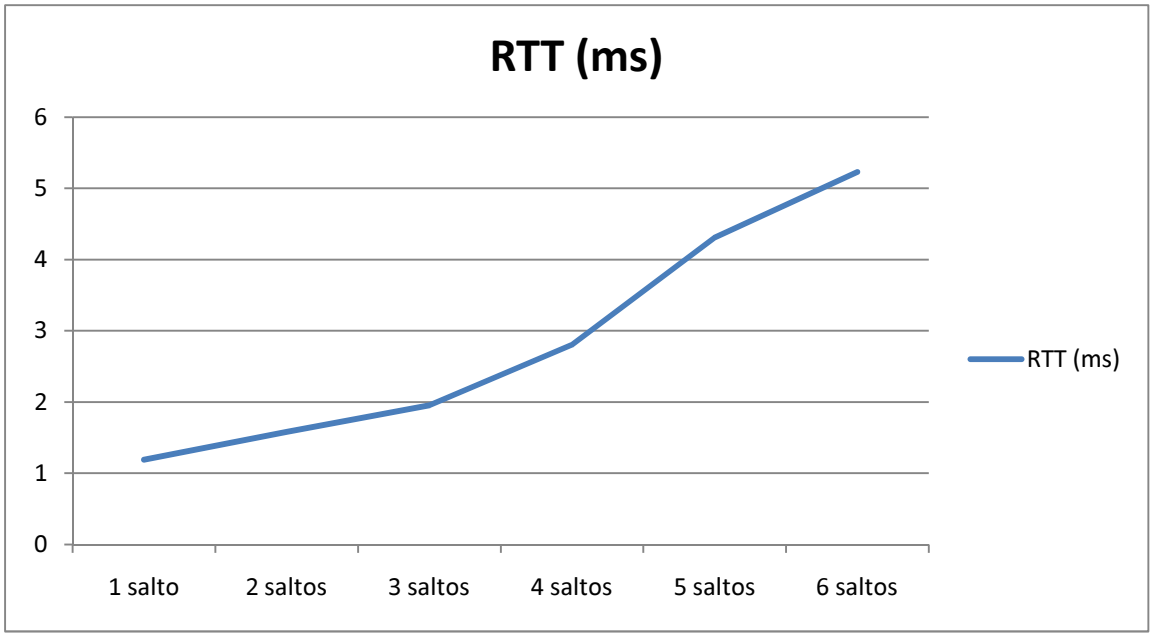

Figura 47 : RTT en función del número de saltos

Lo que cabría esperar es que el tiempo RTT creciera de forma lineal con el número de saltos entre origen y destino debido a que en cada salto, se añade el tiempo que un nodo tarda en procesar y redirigir el paquete. De esta manera, si el tiempo que un nodo tarda en procesar un paquete se considera constante, el tiempo RTT seguiría la siguiente expresión:

$$RTT = tiempo_{ida} + tiempo_{vuelta} = tiempo_{procesamiento} \cdot ((Nºsaltos_{ida} + 1) + (Nºsaltos_{vuelta} + 1))$$

Como se puede apreciar en la figura 47, el tiempo RTT no crece exactamente como una recta con el número de saltos entre origen y destino. Esto puede deberse a la combinación de los siguientes efectos:

- Como las pruebas se realizan sobre un único ordenador utilizando un software de virtualización, es posible que en ciertos instantes el sistema se encuentre sobrecargado por tener que realizar muchas tareas o tener que acceder a disco duro. Este hecho puede provocar variaciones de tiempo de procesamiento de los paquetes en los diferentes nodos que tengan como consecuencia la no linealidad del tiempo RTT.

- En la implementación se simula el servicio de localización con una tabla en la que cada nodo guarda la posición geográfica de todos los nodos de la red VANET y que es consultada para obtener la posición geográfica del destino ante un envío *geounicast*. Esta tabla se encuentra configurada de forma estática al inicio de cada prueba de forma que, el orden de aparición de los nodos en la tabla, hace que se tarde más tiempo en obtener los datos para los vecinos que se encuentran más lejos de la RSU. Esto provoca la pérdida de linealidad del tiempo RTT con el número de saltos.

El tiempo que tarda un paquete *geounicast* desde la RSU a un vehículo en función del número de saltos es un parámetro muy significativo. En el caso en el que un vehículo de la red VANET establece una comunicación con un *host* de internet a través de la RSU, este tiempo se puede

entender como el tiempo extra que hay que añadir por estar accediendo a internet desde una red VANET y tener como *router* de acceso a la RSU.

Por otro lado, hay que tener en cuenta la influencia del entorno de pruebas sobre esta medida de tiempo. El nivel de enlace considerado en las pruebas es un nivel *Ethernet* en lugar del utilizado en un caso real, que sería el de alguna tecnología radio *wireless* (IEEE 802.11p, IEEE 802.11 a/b/g/n). El hecho de utilizar una red cableada provoca que los retardos de acceso al medio sean despreciables, mientras que, en las tecnologías *wireless*, el retardo de acceso al medio es un parámetro muy influyente en el tiempo RTT. Por ello, proponemos que en trabajos futuros se realicen este tipo pruebas en escenarios en el que se utilicen tecnologías radio *wireless* en lugar de una red cableada para obtener valores más realistas.

6.2.2- Segundo escenario de pruebas: medida del retardo de paquetes *geobroadcast* desde la RSU a una OBU y del tiempo de configuración de una dirección IPv6

Con este escenario de pruebas se pretende:

- Validar el funcionamiento de la implementación para el envío de paquetes *geobroadcast* entre nodos de la red VANET.

- Comprobar el tiempo que tarda una OBU en configurar su dirección IPv6 cuando entra en una nueva zona geográfica a cargo de una nueva RSU.

Se han reproducido las pruebas realizadas en el documento [16] con la intención de tener una base con la que validar la implementación realizada. Los resultados de este documento se obtuvieron en los laboratorios de NEC [33] empleando su propia implementación de encaminamiento geográfico y utilizando como nivel de enlace IEEE 802.11a. El escenario de la prueba se puede observar en la figura 48.

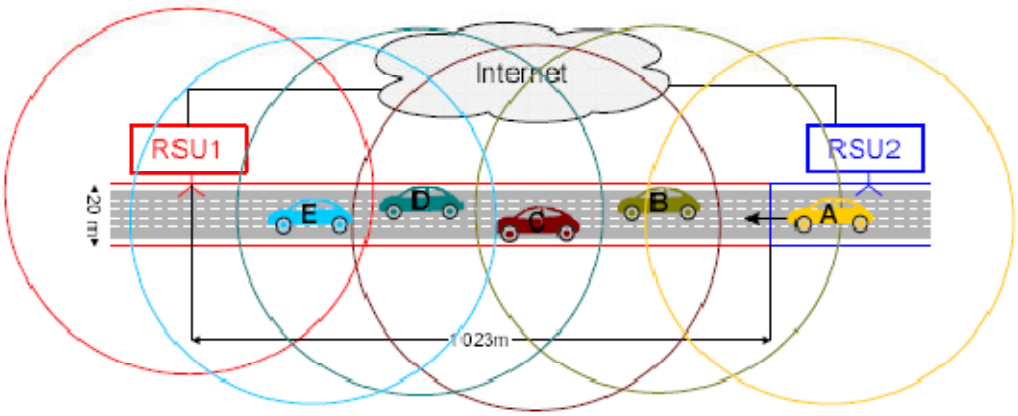

Figura 48 : Segundo escenario de pruebas: medida del retardo de paquetes *geobroadcast* desde la RSU a una OBU y del tiempo de configuración de una dirección IPv6

En este escenario, la RSU1 y RSU2 difunden paquetes *Router Advertisement* dentro de su área geográfica asociada. Estas zonas geográficas tienen forma rectangular cuyas medidas son de 20 metros por 1023 metros. Los rectángulos se han dispuesto de tal manera que cubran una carretera sin que exista solapamiento entre ellos. Por lo tanto, la separación entre ambas RSUs es de 2046 metros.

Por otro lado, en el escenario se tienen 5 vehículos dispuestos en línea recta. De ellos, el único que se encuentra en movimiento es el vehículo A, que se dirige de la zona gobernada por la RSU2 a la zona de la RSU1 con una velocidad de 45 Km/h. En el instante de inicio, el vehículo A se encuentra en un punto geográfico que se escoge de forma aleatoria. Este punto geográfico se obtiene considerando que el vehículo A se coloca a una distancia d_inicial de la RSU2 siguiendo una distribución uniforme. De esta forma el punto queda distribuido uniformemente entre la RSU2 y la frontera de su área de influencia.

El resto de vehículos (B, C, D y E) no se mueven y se encuentran dentro del área de influencia de la RSU1. Estos vehículos se encuentran equiespaciados con una distancia entre ellos de 209 metros. Esta distancia ha sido escogida intencionadamente para que un vehículo sólo tenga comunicación directa con otros dos. De esta manera, cuando el vehículo A entre en el área de la RSU1 podrá configurar una dirección IPv6 al recibir un paquete *Router Advertisement*. Este paquete, habrá sido difundido mediante un envío *geobroadcast* dentro el área de la RSU1, pero que, por la disposición de los nodos en la red, deberá ser reenviado por todos los vehículos hasta que llegue al vehículo A de manos del vehículo B. El rango de cobertura de los nodos de la red es de 250 metros.

Para generar las posiciones iniciales de los nodos de la red se ha utilizado un programa en lenguaje C llamado generador_posiciones. Este programa, que se ejecuta en la máquina virtual especial, genera los ficheros de posiciones iniciales siguiendo las premisas del escenario y los almacena en el directorio /files/. De esta manera, las máquinas virtuales estándar pueden saber su posición inicial. La utilización de este programa se describe en el apéndice C.

En este escenario se pueden realizar dos medidas de tiempo:

- Tiempo que tarda un paquete *geobroadcast* desde la RSU1 hasta el vehículo A tras ser retransmitido salto a salto por el resto de vehículos. A este tiempo le llamamos tiempo de retransmisión.

- Tiempo que transcurre desde que el vehículo A entra en la zona geográfica gobernada por la RSU1 hasta que puede configurar una dirección IPv6. A este tiempo le llamamos tiempo de configuración.

Para que una máquina virtual estándar pueda escribir en fichero los resultados de la prueba, se han aumentado los tipos de peticiones y campos de mensaje del protocolo de comunicación entre el módulo del *kernel* y el proceso auxiliar de espacio de usuario mencionados en el apartado 5.3.1.

Los dos nuevos tipos de peticiones introducidos son:

- Guardar el resultado del vehículo móvil

  Con este tipo de mensaje, el módulo del *kernel* solicita al proceso auxiliar que guarde en el fichero *resultado_movil.txt* los resultados de la medida. La información guardada, en el orden que aparece en el fichero, es:

  - Marca de tiempo del paquete *geobroadcast* del *Router Advertisement* de la RSU1.

  - Distancia recorrida por el vehículo desde el comienzo de la ejecución.

  - Longitud geográfica actual del vehículo.

  - Tiempo actual, es decir, el instante en el que se recibe el *Router Advertisement* de la RSU1.

- Guardar el resultado de la RSU

  Con este tipo de mensaje, el módulo del *kernel* solicita al proceso auxiliar que guarde en el fichero *resultado_rsuX.txt* los resultados de la medida. La información guardada, en el orden que aparece en el fichero, es:

  - Marca de tiempo del paquete *geobroadcast* del *Router Advertisement* que genera la RSU.

  - Tiempo actual, es decir, el instante de tiempo en el que la RSU transmite el paquete *Router Advertisement*.

Los nuevos campos introducidos en el mensaje intercambiado entre el módulo del *kernel* y el proceso de espacio de usuario son:

- *TS*: es la marca de tiempo de los paquetes *geobroadcast* que se desea guardar en fichero.

- *Sec*: son los segundos del instante actual.

- *Usec:* son los microsegundos del instante actual.

Una vez que se termina la prueba, se obtienen los resultados de tiempo de retransmisión y tiempo de configuración a partir de los datos anteriores. Para ello se utiliza un programa en lenguaje C llamado analisis_resultados que analiza los ficheros generados por los nodos. Los resultados de tiempo de configuración y tiempo de retransmisión se guardan en los ficheros *resultado_conf.txt* y *resultado_relay.txt* respectivamente. La utilización de este programa se describe en el apéndice C.

Para poder repetir de forma automática la toma de muestras, se utilizaron unos *scripts* de *Shell* de Linux que se encargaban de ello. Estos *scripts* se pueden ver en el apéndice C. El número de repeticiones de cada medida ha oscilado entre 150 veces en el peor de los casos y 380 veces en el mejor.

A continuación se presentan los resultados obtenidos para las medidas de tiempo realizadas.

6.2.2.1- Tiempo de retransmisión

El tiempo medio que tarda un paquete *geobroadcast* desde la RSU1 hasta el vehículo A tras ser retransmitido salto a salto por el resto de vehículos es de 8,403 milisegundos. El documento con el que comparamos [16] otorga un resultado de 7,026 milisegundos por lo que podemos decir que la implementación no tiene un funcionamiento especialmente lento en lo que al envío de paquetes *geobroadcast* se refiere. Además, teniendo en cuenta que en [16] se utiliza IEEE 802.11a como nivel de enlace, podemos decir que, bajo la condición de ausencia de colisiones que provoquen retardos de acceso al medio (medio de comunicación óptimo y baja utilización de canal), nuestros resultados utilizando máquinas virtuales y nivel de enlace *Ethernet* para crear el entorno de pruebas, son comparables a los de un entorno de pruebas real y una tecnología *wireless* como nivel de enlace.

6.2.2.2- Tiempo de configuración

El tiempo de configuración depende del intervalo tiempo que transcurre entre dos *Router Advertisement* consecutivos. El tiempo entre dos *Router Advertisement* se distribuye uniformemente entre dos valores de tiempo, T_{RAmin} y T_{RAmax}. En la tabla 5 se muestran los resultados obtenidos para diferentes intervalos de tiempo entre paquetes *Router Advertisement*.

Para comparar, también se presentan los resultados del documento de referencia [16].

T_{RAmin}(s)	T_{RAmax}(s)	$\overline{T_{configuración}}(s)$	$\overline{T_{configuración\ [16]}}$(s)
0,3	0,4	0,175	0,178
0,4	0,6	0,246	0,258
0,8	1,2	0.518	0,511
1,6	2,4	1,068	1,017
4	6	2.436	2,535

Tabla 5 : Resultados del tiempo de configuración

Como se puede observar comparando los resultados de la implementación con los del documento [16], se obtienen tiempos de configuración muy similares. Esto indica la concordancia entre los resultados prácticos de nuestra implementación y el análisis teórico y práctico del tiempo de configuración realizado en el documento [16] con el que comparamos.

Capítulo 7

Conclusiones y trabajos futuros

Las conclusiones que podemos obtener de la evaluación de la implementación son:

- Se ha conseguido el primer gran objetivo del proyecto: la implementación de un protocolo de encaminamiento geográfico que permita el encaminamiento de paquetes basándose en la posición geográfica de los nodos que forman la red. Se han conseguido realizar los dos tipos de envíos de paquetes que se pretendían: entrega de paquetes *geounicast* y entrega de paquetes *geobroadcast*.

- También se ha llegado al segundo gran objetivo: la integración de IPv6 sobre el protocolo de encaminamiento geográfico. Se han realizado los cambios necesarios sobre el nivel IPv6 para que las OBUs de la red vehicular puedan configurar la dirección IPv6 de sus interfaces de manera automática cuando reciben un paquete *Router Advertisement* de la RSU. Del mismo modo, los vehículos de la red son capaces de soportar IPv6 y así, poder comunicarse con otros nodos utilizando IPv6.

- Se han obtenido unos resultados muy similares al reproducir las pruebas realizadas por los laboratorios de NEC en la que se emplea otra implementación de encaminamiento geográfico y se utiliza como nivel de enlace IEEE 802.11a. Esto, además de validar el funcionamiento de la implementación en cuanto al envío de paquetes *geobroadcast* y configuración automática de las direcciones IPv6 de las OBUs, demuestra que bajo condiciones de medio de comunicación óptimo y baja utilización de canal (ausencia de colisiones que provoquen retardos de acceso al medio), *VMWare Server 2.0* proporciona un entorno de pruebas basado en máquinas virtuales interconectadas por *Ethernet* con el que se obtienen resultados comparables a los de un escenario real con una tecnología *wireless* como nivel de enlace.

- El tiempo RTT del envío de paquetes entre la RSU y una OBU en el peor caso considerado (6 saltos) es de 5,228 milisegundos. Este valor de tiempo es aceptable para poder establecer comunicaciones entre nodos de la red VANET o entre éstos y un *host* de internet a través de la RSU.

- *Click Modular Router* nos ha permitido desarrollar una implementación divida en módulos y flexible en la que es sencillo introducir nuevas funcionalidades con nuevos elementos. Además, con esta herramienta hemos podido ejecutar código en el *kernel* de Linux que sirva de intermediario entre las interfaces de red y la pila de red de Linux.

Por otro lado, nos gustaría plantear diversos aspectos en los que se puede seguir trabajando en el futuro:

- En la implementación, el periodo con el que los nodos de la red VANET envían los paquetes baliza para informar al resto de nodos vecinos de su posición geográfica es de 500 milisegundos. El tiempo de validez de una entrada en la tabla de vecinos es de 10 segundos. Se han adoptado estos valores siguiendo las especificaciones del documento [15], aunque como trabajo futuro, se podrían plantear estudios sobre el efecto de la variación de estos tiempos.

- Para el encaminamiento de los paquetes en la implementación se ha utilizado *greedy forwarding*. Como trabajo futuro se plantea modificar el encaminamiento de paquetes implementado para que se permita una entrega de paquetes más fiable. *Greedy forwarding* tiene problemas cuando un nodo no puede encontrar ningún vecino dentro de su radio de alcance que se encuentre más cerca del destino que él mismo.

- Por otro lado, el protocolo de encaminamiento geográfico implementado emplea inundación simple para hacer llegar un paquete *geobroadcast* a todos los nodos que se encuentran dentro de la región destino. Habría que mejorar el mecanismo de inundación utilizado ya que el de inundación simple es poco eficiente debido al gran número de retransmisiones redundantes innecesarias.

- En el proyecto hemos considerado que las zonas geográficas que cubren las RSUs no se solapan entre sí. Otro aspecto que se podría trabajar en un futuro es considerar el caso en el que estas regiones geográficas sí que comparten zonas en común. De esta manera, se podrían estudiar mecanismos de *handover* más sofisticados, en los que por ejemplo, un vehículo pueda comenzar a configurar una nueva dirección IPv6, mientras que la antigua todavía está en uso.

- También hay que tener en cuenta la influencia del entorno de pruebas sobre las medidas de tiempo realizadas. El nivel de enlace considerado en las pruebas es un nivel *Ethernet* en lugar del utilizado en un caso real, que sería el de alguna tecnología radio *wireless* (IEEE 802.11p, IEEE 802.11 a/b/g/n). El hecho de utilizar una red cableada provoca que los retardos de acceso al medio sean despreciables, mientras que, en las tecnologías *wireless*, el retardo de acceso al medio es un parámetro muy influyente. Por ello, proponemos que en trabajos futuros se realicen pruebas en escenarios en el que se utilicen tecnologías radio *wireless* en lugar de una red cableada.

- Por último, señalar que el servicio de localización y de generación de posición actual de los nodos se encuentran sin completar en la implementación. Por ello, se plantea que en un futuro, se adapte la implementación a la utilización de un servicio de localización y un sistema GPS reales.

Apéndice A

Planificación del proyecto

A continuación se presenta la división en actividades y tareas del proyecto, se describen brevemente sus objetivos y se realiza una planificación temporal de las mismas. Para la planificación temporal se consideran jornadas de 8 horas.

A.1- División en actividades y tareas del proyecto

El proyecto se puede dividir en las siguientes actividades:

A. Documentación y estudio del estado del arte del proyecto

B. Diseño del protocolo de encaminamiento geográfico e integración con IPv6

C. Familiarización con la herramienta *Click Modular Router*

D. Desarrollo de la implementación

E. Despliegue del entorno de evaluación de la implementación

F. Evaluación de la implementación

G. Documentación y memoria del proyecto

Cada una de estas actividades se divide en una serie de tareas:

A- <u>Documentación y estudio del estado del arte del proyecto:</u>

<u>A.1-Búsqueda de bibliografía:</u> el objetivo de esta tarea es encontrar información sobre redes VANET y protocolos de encaminamiento geográfico. La duración de la fase de búsqueda bibliográfica es de 6 jornadas.

<u>A.2-Análisis de la bibliografía:</u> con esta tarea se pretende obtener los conocimientos necesarios para la realización del proyecto. Esta tarea tiene asignada una duración de 10 jornadas.

B- **Diseño del protocolo de encaminamiento geográfico e integración con IPv6:**

B.1-Diseño del protocolo de encaminamiento geográfico: en esta tarea se lleva a cabo la elección del protocolo de encaminamiento geográfico que se va a utilizar y se estudian los campos de las cabeceras de los paquetes. La duración de esta tarea es de 7 jornadas.

B.2- Diseño de la integración de IPv6 sobre el protocolo de encaminamiento geográfico: con esta tarea se pretenden deducir los cambios necesarios en IPv6 para su integración con el nivel de encaminamiento geográfico. Esta tarea se realiza durante 5 jornadas.

C- **Familiarización con la herramienta *Click Modular Router***

C.1- Análisis de la documentación de *Click Modular Router*: el objetivo de la tarea es entender el funcionamiento de la herramienta y poder empezar a trabajar con ella. La duración de esta tarea es de 4 jornadas.

C.2- Instalación de *Click Modular Router*: se trata de configurar un *kernel* válido con el parche de *Click Modular* Router donde poder instalar la herramienta. Para esta tarea son necesarias 6 jornadas de trabajo.

C.3- Realización de configuraciones y elementos sencillos de *Click Modular Router*: con esta tarea se pretende obtener habilidad en el manejo de la herramienta implementando elementos de *Click Modular Router* y aplicándolos en configuraciones sencillas. Se emplean 5 jornadas para ello.

D- **Desarrollo de la implementación**

D.1- División en elementos de la implementación: el objetivo de la tarea es diseñar una división la implementación en elementos de *Click Modular Router*. Son necesarias 2 jornadas para esta tarea.

D.2- Desarrollo de los elementos: se trata de proceder con el desarrollo de cada uno de los elementos en los que se ha dividido la implementación. Además de programar cada elemento, se le realizan pruebas para comprobar su funcionamiento. Esta es la tarea más laboriosa del proyecto y por ello se emplean 25 jornadas de trabajo.

D.3- Programación del proceso auxiliar de nivel de usuario: en esta tarea se implementa el proceso auxiliar con el que se comunica el módulo del *kernel* de Linux y se prueba el funcionamiento de la comunicación. La tarea tiene una duración de 4 jornadas.

D.4- Unión de los elementos y pruebas básicas: con esta tarea se pretende obtener una configuración de *Click Modular Router* que funcione correctamente. Para ello se emplean 4 jornadas.

E- **Despliegue del entorno de evaluación de la implementación**

E.1- Análisis de los software de virtualización: con esta tarea se pretende decidir sobre el software de virtualización que se va a elegir para realizar la evaluación de la implementación. Para ello, es necesario analizar su documentación. Se establecen 2 jornadas para esta tarea.

E.2- Decisión de las pruebas a realizar: el objetivo de esta tarea es planificar y decidir las pruebas que se van a realizar de la implementación. Se emplean 2 jornadas para esta tarea.

E.3- Preparación del entorno de pruebas: se trata de preparar el entorno de pruebas para evaluar la implementación: instalar el software de virtualización, instalar la implementación, preparativos necesarios para las pruebas, etc. La duración de la tarea es de 11 jornadas.

F- **Evaluación de la implementación**

F.1- Realización de las pruebas de la implementación: en esta tarea se prueba la implementación en función de las pruebas que se hayan decido realizar. Son necesarias 7 jornadas para esta tarea.

G- **Documentación y memoria del proyecto**

G.1- Extracción de conclusiones y puntos importantes del proyecto: en esta tarea se extraen las conclusiones y puntos importantes de la realización del proyecto que deben aparecer en la memoria. Se dedican 3 jornadas a esta tarea.

G.2- Planificación de la redacción de la memoria: el objetivo de esta tarea es acordar los apartados en los que se divide la memoria y el contenido de cada uno de ellos. Se emplean 3 jornadas para esta tarea.

G.3- Redacción de la memoria: se redacta la memoria. La duración de la fase de redacción es de 20 jornadas.

G.4- Revisión y corrección de la memoria: se revisa la memoria y se corrigen los fallos. Esta tarea dura 5 jornadas.

A.2- Cuadro resumen de la planificación temporal del proyecto

A continuación se presenta una tabla con el resumen de la planificación temporal del proyecto:

Tarea	Descripción de la tarea	Duración
A	**Documentación y estudio del estado del arte del proyecto**	**16 jornadas**
A1	Búsqueda de bibliografía	6 jornadas
A2	Análisis de la bibliografía	10 jornadas
B	**Diseño del protocolo de encaminamiento geográfico e integración con IPv6**	**12 jornadas**
B1	Diseño del protocolo de encaminamiento geográfico	7 jornadas
B2	Diseño de la integración de IPv6 sobre el protocolo de *routing* geográfico	5 jornadas
C	**Familiarización con la herramienta *Click Modular Router***	**15 jornadas**
C1	Análisis de la documentación de *Click Modular Router*	4 jornadas
C2	Instalación de *Click Modular Router*	6 jornadas
C3	Realización de configuraciones y elementos sencillos de *Click Modular Router*	5 jornadas
D	**Desarrollo de la implementación**	**35 jornadas**
D1	División en elementos de la implementación	2 jornadas
D2	Desarrollo de los elementos	25 jornadas
D3	Programación del proceso auxiliar de nivel de usuario	4 jornadas
D4	Unión de los elementos y pruebas básicas	4 jornadas
E	**Despliegue del entorno de evaluación de la implementación**	**15 jornadas**
E1	Análisis de los software de virtualización	2 jornadas
E2	Decisión de las pruebas a realizar	2 jornadas
E3	Preparación del entorno de pruebas	11 jornadas
F	**Evaluación de la implementación**	**7 jornadas**
F1	Realización de las pruebas de la implementación	7 jornadas
G	**Documentación y memoria del proyecto**	**31 jornadas**
G1	Extracción de conclusiones y puntos importantes del proyecto	3 jornadas
G2	Planificación de la redacción de la memoria	3 jornadas
G3	Redacción de la memoria	20 jornadas
G4	Revisión y corrección de la memoria	5 jornadas
	TOTAL	**139 jornadas**

Tabla 6 : Planificación temporal del proyecto

Según la planificación, la duración total del proyecto es de 139 jornadas de 8 horas de trabajo. Si se consideran 22 jornadas de trabajo al mes, la duración total del proyecto es de 6 meses y 7 jornadas de trabajo.

Apéndice B

Manual de instalación

B.1- Instalación de *Click Modular Router* y de la implementación

A continuación se detallan los pasos de la instalación de *Click Modular Router* y de nuestra implementación en una máquina con sistema operativo Linux Ubuntu 8.04.

Paso 1:

En primer lugar hay que obtener la versión de *Click Modular Router* más reciente. Esto se puede realizar con el comando:

```
bash:$ sudo git clone git://read.cs.ucla.edu/git/click /click
```

Con esto lo que conseguimos es descargar del repositorio el código fuente de *Click Modular router* en el directorio /click de nuestra máquina. Actualmente, la versión más reciente de *Click Modular Router* es la 1.7.Orc1.

Si no disponemos de la herramienta git, podemos instalarla con las siguientes instrucciones:

```
bash:$ sudo apt-get update
bash:$ sudo apt-get install git-core
```

Para los pasos que vienen a continuación se ha seguido la referencia [36] que explica como compilar un *kernel* de Linux en Ubuntu.

Paso 2:

Como para que *Click Modular Router* pueda ejecutarse como un módulo del *kernel* hay que aplicar un parche al *kernel* de Linux, el siguiente paso es obtener el código fuente de un *kernel* al que se le pueda aplicar el parche. En nuestro caso, el *kernel* utilizado es la versión 2.6.24.7. Podemos obtener el archivo tar.bz2 con las fuentes del *kernel* de la página web http://kernel.org/(http://www.kernel.org/pub/linux/kernel/v2.6/linux-2.6.24.7.tar.bz2) [37].Copiamos el archivo en el directorio /usr/src/.

Paso 3:

Para aplicar el parche de *Click Modular Router* al *kernel* y compilarlo vamos a necesitar los paquetes *build-essential* y *kernel-package*. Podemos instalarlos con el comando:

```
bash:$ sudo apt-get update
bash:$ sudo apt-get install build-essential kernel-package
```

Paso 4:

Ahora hay que descomprimir las fuentes del *kernel*. Nos movemos al directorio /usr/src/:

```
bash:$ cd /usr/src
```

Si existe un enlace llamado linux a un *kernel* antiguo, lo borramos:

```
bash:/usr/src$ sudo rm linux
```

Se descomprime el archivo de las fuentes con el comando:

```
bash:/usr/src$ sudo tar jxvf linux-2.6.24.7.tar.bz2
```

Esto creará un directorio llamado linux-2.6.24.7.Creamos un enlace simbólico (llamado linux) a ese directorio:

```
bash:/usr/src$ sudo ln -s linux-2.6.24.7 linux
```

Paso 5:

Es el momento de aplicar el parche al *kernel*. Esto se realiza con los comandos:

```
bash:$ cd /usr/src/linux-2.6.24.7
bash:/usr/src/ linux-2.6.24.7$ sudo patch –p1 -b < /click/etc/linux-
2.6.24.7-patch
```

Paso 6:

Para la integración de IPv6 sobre el nivel de encaminamiento geográfico ha sido necesario introducir unos cambios en los archivos *if_arp.h* y *addrconf.c* del *kernel* de Linux. Por ello es necesario sustituirlos en el código fuente descargado. El archivo *if_arp.h* se encuentra en el directorio /usr/src/linux-2.6.24.7/include/linux/ y el archivo *addrconf.c* en /usr/src/linux-2.6.24.7/net/ipv6/. Sustituimos estos archivos por los suministrados en la implementación.

Paso 7:

Llega uno de los pasos más complicados en la compilación del *kernel*, su configuración. Para poder configurar el *kernel* de manera gráfica, instalamos el paquete *libncurses5-dev*:

```
bash:$ cd /usr/src/
bash:/usr/src$ sudo apt-get install libncurses5-dev
```

Para que la configuración del *kernel* sea más sencilla podemos partir de la configuración del *kernel* actual. Ésta se encuentra en un archivo de texto llamado *config-versión* en el directorio /boot/. Por lo tanto, para partir de esa base y comenzar la configuración del nuevo *kernel* desde la actual, lo que tenemos que hacer es copiar dicho archivo al directorio /usr/src/linux-2.6.24.7/, pero llamándolo .config:

```
bash:$ cd /usr/src/linux-2.6.24.7/
bash:/usr/src/linux-2.6.24.7$ sudo cp /boot/config-versión .config
```

Ahora ya podemos configurar el *kernel* ejecutando:

```
bash:/usr/src/linux-2.6.24.7$ sudo make oldconfig menuconfig
```

En este momento aparecerán varías preguntas sobre la configuración del *kernel*. Lo más seguro es responder con la configuración por defecto.

Por otro lado, hay que prestar atención a que la opción CONFIG_PREEMPT esté deshabilitada. Para ello, la opción CONFIG_PREEMPT_VOLUNTARY debe ser activada. De otra manera *Click Modular Router* no funcionará correctamente.

Paso 8:

El paso siguiente es construir el paquete con el que instalaremos el nuevo *kernel*. Ejecutamos el comando:

```
bash:/usr/src/linux-2.6.24.7$ sudo make-kpkg clean
bash:/usr/src/ linux-2.6.24.7$ sudo make-kpkg --append-to-
version=.XXXX --initrd kernel_image
```

Donde XXXX representa una secuencia alfanumérica para distinguir diferentes versiones de compilación del *kernel*. Por ejemplo, se puede sustituir XXXX por la fecha de la versión o su número. Como sólo vamos a realizar una versión de compilación del *kernel*, la orden quedaría así:

```
bash:/usr/src/linux-2.6.24.7$ sudo make-kpkg --append-to-version=.1 --
initrd kernel_image
```

El proceso de creación del paquete dura bastante tiempo.

Paso 9:

Una vez que se ha creado el paquete, éste se encuentra ubicado en el directorio /usr/src/. Para instalarlo ejecutamos:

```
bash:$ cd /usr/src/
bash:/usr/src$ sudo dpkg -i linux-image-2.6.24.7.1.Custom_i386.deb
```

Paso 10:

Ya tenemos el *kernel* listo para instalar *Click Modular Router*. Sólo nos falta reiniciar el ordenador y seleccionar nuestro *kernel* en el gestor de arranque *grub*.

Paso 11:

Antes de compilar *Click Modular Router*, es necesario:

- Copiar los elementos de nuestra implementación en el directorio /click/elements/local/. Esto sirve para que *Click Modular Router* tenga en cuenta los nuevos elementos que hemos implementado.

- Modificar la definición de la macro *IS_RSU* (línea 21 del fichero *definitions.hh*) en función del comportamiento que queramos que tenga el nodo, es decir, RSU u OBU.
- Ejecutar los siguientes comandos para preparar *Click Modular Router* para la compilación:

```
bash:$ cd /click/
bash:/click$ sudo ./configure --enable-local --enable-ip6 --
disable-userlevel.
```

Con la opción *--enable-local* conseguimos que se compilen los elementos que se encuentran en ese directorio. Con la opción *--enable-ip6* conseguimos que se compilen los elementos de IPv6. Con la opción *--disable-userlevel* compilamos los elementos de *Click Modular Router* para ejecutarlo como un módulo del *kernel*.

Paso 12:

Compilamos *Click Modular Router* ejecutando:

```
bash:$ cd /click/
bash:/click$ sudo make install
```

Esto crea dos archivos objeto: *click.ko* y *proclikefs.ko*. Estos archivos son los que se necesitan para instalar una configuración de *Click Modular Router*.

Paso 13:

Por último, debemos compilar los ficheros del proceso auxiliar que se ejecuta a nivel de usuario y que se comunica con el módulo del *kernel* (hay que tener en cuenta que hay que definir la ubicación de los ficheros de posición inicial, versión, etc. en el archivo *auxiliar_process.c* antes de la compilación). Para la compilación, nos situamos en el directorio donde éste se encuentre y ejecutamos el comando:

```
bash:/directorio_proceso_auxiliar$ gcc –lm –o proceso_auxiliar
auxiliar_process.c restaTiempos.c
```

Las instrucciones para instalar la configuración de nuestra implementación como un módulo del *kernel* se encuentran en el apéndice C.

B.2- Instalación de *Router ADVertisement Daemon (RADVD)*

En aquellos nodos de la red que vayan a actuar como RSU es necesaria la instalación de este demonio que se encarga de enviar *Router Advertisement*, tal y como se tratara de un *router* IPv6.

Para instalar este demonio basta con ejecutar los siguientes comandos:

```
bash:$ sudo apt-get update
bash:$ sudo apt-get install radvd
```

Para su correcta utilización es necesaria una pequeña configuración:

- Se debe habilitar en reenvío de paquetes IPv6. Para ello, basta ejecutar la instrucción:

```
bash:$ sudo sysctl -w net.ipv6.conf.all.forwarding=1
```

- El demonio se configura mediantes el fichero */etc/radvd.conf*. El archivo de configuración utilizado para una RSU es el siguiente:

```
interface fake0 {
        AdvSendAdvert on;
        MinRtrAdvInterval 4;
        MaxRtrAdvInterval 6;
        prefix 2001:1:1:1::/64 {
                AdvOnLink on;
                AdvAutonomous on;
                AdvRouterAddr on;
        };
};
```

  Donde *fake0* es la interfaz de red por la que se envían los paquetes. *AdvSendAdvert=on* establece que se active el envío de *Router Advertisement*. *MinRtrAdvInterval* y *MaxRtrAdvInterval* son los intervalos de tiempo mínimo y máximo entre dos paquetes *Router Advertisement*. De esta manera, el intervalo de tiempo entre el envío de dos *Router Advertisement* consecutivos, se distribuye uniformemente entre estos valores. Por último, el grupo de sentencias *prefix {}* indica el prefijo difundido y las opciones del paquete *Router Advertisement*. Para más información sobre el archivo de configuración se puede ejecutar:

```
bash:$ man radvd.conf
```

Las instrucciones de cómo utilizar el demonio con nuestra implementación se encuentran en el apéndice C.

B.3- Instalación de *Precision Time Protocolo daemon (PTPd)*

A la hora de evaluar la implementación, si se desean realizar medidas de tiempo que vinculen a más de un nodo de la red es necesario sincronizar sus relojes. Para ello, se ha utilizado el demonio *PTPd*.

Para su instalación se debe obtener el código fuente del demonio de la página web http://ptpd.sourceforge.net/ [32]. El fichero con la versión más reciente se llama *ptpd-1.0.0.tar.gz.*

La instalación es muy sencilla:

- Se descomprime el fichero con el código fuente en un directorio. Por ejemplo, vamos a descomprimirlo en el directorio raíz:

```
bash:$ cd /
bash:/$ sudo tar xvf ptpd-1.0.0.tar.gz
```

- Se compila el código fuente:

```
bash:$ cd /ptpd-1.0.0/src/
bash:/ptpd-1.0.0/src$ sudo make
```

Una vez instalado el programa hay que iniciar su ejecución con los comandos:

```
bash:$ cd /ptpd-1.0.0/src/
bash:/ptpd-1.0.0/src$ sudo ./ptpd –b interfaz
```

Donde *interfaz* es el dispositivo de red que se va a utilizar para sincronizarse con el resto de máquinas.

El demonio *ptpd* debe encontrarse en ejecución en todas aquellas máquinas que se deseen sincronizar.

B.4- Instalación del servidor y cliente *NFS (Network File System)*

A la hora de evaluar la implementación, es necesario instalar un servidor NFS en la máquina virtual especial para que las máquinas virtuales estándar puedan montar, utilizando un cliente NFS, el directorio /files/ donde se encuentran los ficheros de posición inicial y resultados.

Para la instalación del servidor y cliente NFS se ha seguido la referencia [38].

Instalación del servidor NFS

Para instalar el servidor NFS debemos ejecutar los siguientes comandos:

```
bash:$ sudo apt-get update
bash:$ sudo apt-get install nfs-kernel-server nfs-common portmap
```

Esto instalará los paquetes que contienen el servidor NFS.

Para configurar el servidor NFS para que exporte el directorio /files/ debemos editar el fichero */etc/exports* añadiendo la siguiente línea:

```
/files 192.168.0.0/24 (rw,no_root_squash,sync)
```

Esto permite a las máquinas de la red 192.168.0.X montar el directorio /files.

Para que se aplique esta configuración reiniciamos el servidor:

```
bash:$ sudo /etc/init.d/nfs-kernel-server restart
```

Instalación del cliente NFS

Para instalar el cliente NFS debemos ejecutar los siguientes comandos:

```
bash:$ sudo apt-get update
bash:$ sudo apt-get install nfs-common portmap
```

Esto instalará los paquetes que contienen el cliente NFS.

Para montar el directorio /files/ en una máquina estándar hay que ejecutar el siguiente comando:

```
bash:$ sudo mount direccion_ip_maquina_especial:/files/ /files/
```

Apéndice C

Ejecución de la implementación

C.1- Cómo ejecutar la implementación en un nodo que se comporta como una OBU

A continuación se detallan los pasos de la ejecución de la implementación en los nodos que se comportan como una OBU.

Paso 1:

En primer lugar hay que establecer el valor correcto de los ficheros de posición inicial, versión, prioridad, identificador de nivel de encaminamiento geográfico del nodo y potencia de transmisión. El formato de estos ficheros se explicó en el apartado 5.3.1.

Paso 2:

En segundo lugar hay que instalar el archivo de configuración de la implementación como un módulo del *kernel*. En la primera línea del archivo de configuración es necesario especificar la interfaz de red que se va a utilizar y su MAC. Para instalar la configuración se utiliza el comando:

```
bash:$ sudo click-install archivo_configuración.click
```

Paso 3:

El tercer y último paso, es arrancar el proceso auxiliar que se ejecuta en entorno de usuario. Para ello, situados en el directorio donde éste se encuentra, ejecutamos el comando:

```
bash:/directorio_proceso_auxiliar$ ./proceso_auxiliar
```

Con estos tres pasos, la implementación ya se encuentra en ejecución. Si se desea parar la ejecución se debe, primero detener la ejecución del proceso auxiliar de nivel de usuario con la combinación de teclas Control+C, y después, desinstalar el módulo del *kernel* con el comando:

```
bash:$ sudo click-uninstall
```

C.2- Cómo ejecutar la implementación en un nodo que se comporta como una RSU

A continuación se detallan los pasos de la ejecución de la implementación en los nodos que se comportan como una RSU.

Paso 1:

En primer lugar hay que establecer el valor correcto de los ficheros de posición inicial, versión, prioridad, identificador de nivel de encaminamiento geográfico del nodo y potencia de transmisión. El formato de estos ficheros se explicó en el apartado 5.3.1.

Paso 2:

En segundo lugar hay que instalar el archivo de configuración de la implementación como un módulo del *kernel*. En la primera línea del archivo de configuración es necesario especificar la interfaz de red que se va a utilizar y su MAC. Para instalar la configuración se utiliza el comando:

```
bash:$ sudo click-install archivo_configuración.click
```

Paso 3:

El tercer paso, es arrancar el proceso auxiliar que se comunica con el módulo del *kernel*. Para ello, situados en el directorio donde éste se encuentra, ejecutamos el comando:

```
bash:/directorio_proceso_auxiliar$ ./proceso_auxiliar
```

Paso 4:

Por último, se debe configurar la dirección IPv6 de la RSU y arrancar el demonio *radvd* para mandar los paquetes *Router Advertisemet*. Las instrucciones que hay que ejecutar son las siguientes:

```
bash:$ sudo ifconfig fake0 add  dirección_ipv6_RSU
bash:$ sudo ip -6 route add prefijo_dirección_ipv6_RSU/64 dev fake0
bash:$ sudo sysctl -w net.ipv6.conf.all.forwarding=1
bash:$ sudo /etc/init.d/radvd start
```

Con estos pasos, la implementación ya se encuentra en ejecución. Si se desea parar la ejecución se debe, primero detener la ejecución del proceso auxiliar de nivel de usuario con la combinación de teclas Control+C, y después, desinstalar el módulo del *kernel* con el comando:

```
bash:$ sudo click-uninstall
```

C.3- Funcionamiento del programa generador_posiciones

El programa generador_posiciones se encarga de generar las posiciones iniciales de los nodos de la red para los escenarios de pruebas. Los ficheros de posiciones iniciales se almacenan en el directorio /files/. Para que las máquinas virtuales estándar puedan acceder a estos ficheros, deben montar el directorio con el comando:

```
bash:$ sudo mount direccion_ip_maquina_especial:/files/ /files/
```

Para poder utilizar el programa generador_posiciones es necesario compilar su código. Para ello, se ejecuta el siguiente comando en el directorio donde se encuentra ubicado el código fuente:

```
bash:/directorio_generador_posiciones$ gcc –lm –o generador_posiciones
generador_posiciones.c
```

La forma de ejecutar el generador de posiciones para obtener los ficheros de posiciones iniciales del primer escenario de pruebas (apartado 6.2.1) es la siguiente:

```
bash:/directorio_generador_posiciones$ sudo ./generador_posiciones –m1
```

Donde *–m1* indica al programa que se quieren generar los ficheros del primer escenario de pruebas.

La forma de ejecutar el generador de posiciones para obtener los ficheros de posiciones iniciales del segundo escenario de pruebas (apartado 6.2.2) es la siguiente:

```
bash:/directorio_generador_posiciones$ sudo ./generador_posiciones –m2
–vVELOCIDAD –dD_RSU
```

Donde *–m2* indica al programa que se quieren generar los ficheros del segundo escenario de pruebas, VELOCIDAD es la velocidad, expresada en Km/h con la que se desplaza el vehículo A y D_RSU es la distancia, expresada en metros, que separa a las RSUs. Para el caso concreto de la evaluación de la implementación realizado, la velocidad del vehículo A es de 45 Km/h y la distancia entre RSU1 y RSU2 es de 2046 metros.

C.4- Funcionamiento del programa analisis_resultados

El programa analisis_resultados se encarga de obtener los resultados de tiempo de retransmisión y tiempo de configuración del segundo escenario de pruebas (apartado 6.2.2). Para ello, toma los ficheros con las medidas generados por los nodos del directorio /files/, los analiza y guarda los resultados de tiempo de configuración y tiempo de retransmisión los ficheros *resultado_conf.txt* y *resultado_relay.txt* respectivamente. Estos ficheros también se encuentran en el directorio /files/.

Para poder utilizar el programa es necesario compilar su código. Para ello, se ejecuta el siguiente comando en el directorio donde se encuentra ubicado el código fuente:

```
bash:/directorio_analisis_resultdos$ gcc –lm –o analisis_resultados
analisis_resultados.c
```

La forma de ejecutar el analizador de resultados es la siguiente:

```
bash:/directorio_analisis_resultados$ sudo ./analisis_resultados –
dD_RSU
```

Donde D_RSU es la distancia, expresada en metros, que separa a las RSUs. Para el caso concreto de la evaluación de la implementación realizado, la distancia entre RSU1 y RSU2 es de 2046 metros.

C.5- *Scripts* de *Shell* de Linux para la repetición automática de la toma de medidas en el segundo escenario de pruebas

Para la repetición automática de la toma de de medidas en el segundo escenario de pruebas se han utilizado dos *scripts de Shell* de Linux, uno en la máquina virtual especial y otro en la máquina virtual estándar que actúa como el vehículo A. La misión de estos *scripts*, además de la repetición automática, es la de sincronizar el comienzo de ejecución de la prueba en el instante indicado por la variable `mBeginDate` del *script*. De esta manera, se consigue que la máquina virtual estándar que actúa como el vehículo A obtenga su fichero de posición inicial una vez que éste ya ha sido generado.

El *script* de la máquina virtual especial se muestra a continuación:

```
#!/bin/bash
Cont=1
#Tipos de las variables
typeset -i mCurrDate
typeset -i mBeginDate
typeset -i diferencia
typeset -i Intervalo
#Intervalo entre dos ejecuciones automaticas
Intervalo=400
#Instante del comienzo de la ejecucion
mBeginDate='200911100930'
mCurrDate=`date +"%Y%m%d%H%M"`  ## YYYYMMDDHHMM
while [ ${mCurrDate} -lt ${mBeginDate} ]
do
mCurrDate=`date +"%Y%m%d%H%M"`  ## YYYYMMDDHHMM
done

while [ $Cont ]
do
#Comienzo
START=$(date +%s)
#Se borran las medidas de la prueba anterior y se generan los nuevos
#ficheros de posicion
sudo rm /files/resultado_movil.txt
```

```
sudo rm /files/resultado_rsu1.txt
sudo rm /files/resultado_rsu2.txt
sudo generador_posiciones -m1 -v45 -d2046
sudo chmod 777 /files/*
sleep 300
#Se espera a obtener las medidas para obtener los resulatos
sudo analisis_resultados -d2046

#Se espera hasta el siguiente intervalo de tiempo para volver a
#ejecutar la prueba
END=$(date +%s)
DIFF=$(( $END - $START ))
diferencia=${DIFF}
while [ ${diferencia} -le ${Intervalo} ]
do
END=$(date +%s)
DIFF=$(( $END - $START ))
diferencia=${DIFF}
done
done
```

El *script* de la máquina virtual estándar que actúa como el vehículo A es:

```
#!/bin/bash
Cont=1
#Tipos de las variables
typeset -i mCurrDate
typeset -i mBeginDate
typeset -i diferencia
typeset -i Intervalo
#Intervalo entre dos ejecuciones automaticas
Intervalo=400
#Instante del comienzo de la ejecucion
mBeginDate='200910301620'
mCurrDate=`date +"%Y%m%d%H%M"`  ## YYYYMMDDHHMM
while [ ${mCurrDate} -lt ${mBeginDate} ]
do
mCurrDate=`date +"%Y%m%d%H%M"`  ## YYYYMMDDHHMM
done

while [ $Cont ]
do
#Comienzo
START=$(date +%s)
#Se espera a que la maquina virtual especial genere el fichero de
#posicion inicial
sleep 60
sudo click-install /home/vsandonis/pruebas/geonet03.click &
sleep 30
sudo proceso_auxiliar &
sleep 180
#Se termina la prueba
sudo killall -SIGINT proceso_auxiliar
sleep 5
sudo click-uninstall
sleep 5

#Se espera hasta el siguiente intervalo de tiempo para volver a
#ejecutar la prueba
END=$(date +%s)
DIFF=$(( $END - $START ))
```

```
diferencia=${DIFF}
while [ ${diferencia} -le ${Intervalo} ]
do
END=$(date +%s)
DIFF=$(( $END - $START ))
diferencia=${DIFF}
done
done
```

Apéndice D

Archivo de configuración y esquema de elementos de la implementación

El archivo de configuración de la implementación es el siguiente:

```
define($DEVICE eth0, $BEACON_INTERVAL 0.5, $ETHERTYPE 0x0707,$MAC
MAC_DEL_NODO,$RADIO 250)
//Cola
q::Queue;

//Sección que genera los paquetes baliza
sch_beacon::SetCommonHeaderBeacon;
TimedSource($BEACON_INTERVAL)
->sch_beacon
->EtherEncap($ETHERTYPE, $MAC, FF:FF:FF:FF:FF:FF)
->q;


//Sección que procesa los paquetes procedentes del host
sdp::SetDestParam();
ipgeoND::IPV6geoND();
ssPV_unicast::SetSourcePV();
ssPV_broadcast_circular::SetSourcePV();
ssPV_broadcast_rectangular::SetSourcePV();
sch_data::SetCommonHeader;
GeoFromHost(fake0)
->c::Classifier(0/6? 6/3A 40/87,
             0/6?,
             -);
c[0]->Discard;
c[2]->Discard;
c[1]->ipgeoND;
ipgeoND[0]->[0]sdp;
ipgeoND[1]->[1]sdp;

sdp[0]->ssPV_unicast;
sdp[1]->ssPV_broadcast_circular;
sdp[2]->ssPV_broadcast_rectangular;
sdp[3]->Discard;

ssPV_unicast->[0]sch_data;
ssPV_broadcast_circular->[1]sch_data;
ssPV_broadcast_rectangular->[2]sch_data;


//Sección que procesa los paquetes procedentes de otros nodos
//junto con los que vienen del host
mhsbeacon::MultihopSimulation(RADIUS $RADIO);
mhsunicast::MultihopSimulation(RADIUS $RADIO);
```

```
mhsbroadcast::MultihopSimulation(RADIUS $RADIO);
ngh::NextGeoHop(ETH $MAC);
cch::ChangeCommonHeader();
chip::MarkIP6Header(OFFSET 14);
FromDevice($DEVICE)->classif::Classifier(12/0707 15/1?,
             12/0707 15/2?,
        12/0707 15/4?,
             -);
//Descarte de los paquetes procedentes de nodos fuera de cobertura
mhsbeacon[1]->Discard;
mhsunicast[1]->Discard;
mhsbroadcast[1]->Discard;

//Paquetes de otros nodos
classif[0]->mhsbeacon[0]->[0]ngh;
classif[1]->mhsunicast[0]->[1]ngh;
classif[2]->mhsbroadcast[0]->[2]ngh;
classif[3]->Discard;
//Paquetes del host
sch_data[0]->[3]ngh;
sch_data[1]->[4]ngh;

//Paquetes dirigidos al host
ngh[0]->EraseGeoHeader->EtherEncap(0x86DD,ff:ff:ff:ff:ff:ff, $MAC)
->SetPacketType(HOST)
->chip
->ToHost(fake0);

//Paquetes hacia otros nos
ngh[1]->cch[0]->q;
ngh[2]->q;
ngh[3]->Discard;
cch[1]->Discard;

//A la tarjeta de red
q->ToDevice($DEVICE,NO_PAD true);
```

El esquema del grafo en el que se traduce este archivo de configuración es el de la figura 49.

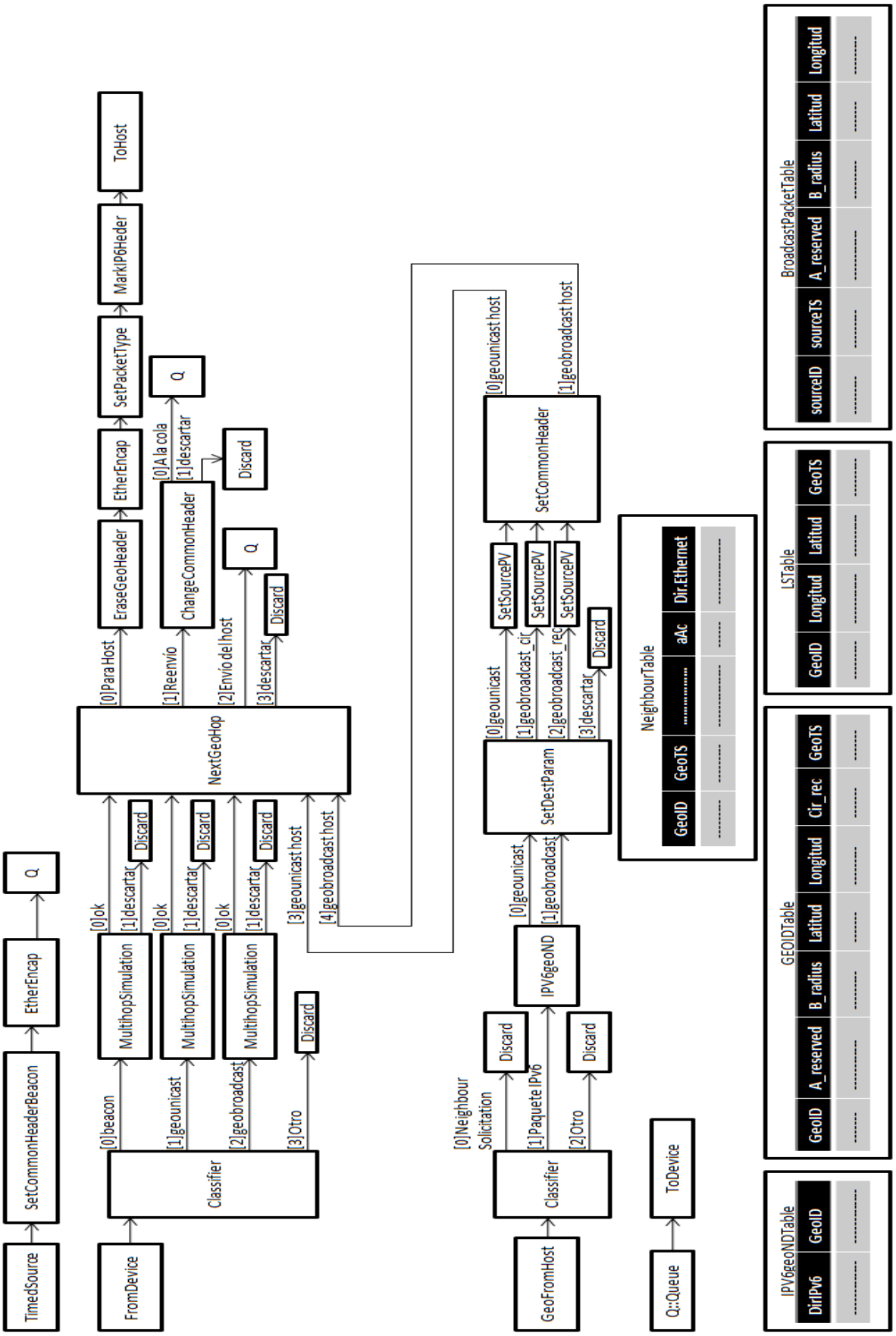

Figura 49 : Esquema de elementos de la implementación

Apéndice E

Contenido del CDROM

Con la memoria se adjunta un CDROM que contiene, además de la memoria en formato PDF, el código de la implementación y de las pruebas realizadas en el proyecto fin de carrera.

En la raíz del CDROM tenemos el directorio llamado "PFC Víctor Sandonís Consuegra". Dentro de este directorio podemos encontrar:

- Subdirectorio "Código_implementación": este subdirectorio se divide a su vez en otros cuatro subdirectorios:
    - "configuración_click": donde podemos encontrar el archivo de la configuración de la implementación.
    - "configuración_radvd": donde se encuentra el archivo de configuración del demonio RADVD utilizado para enviar paquetes *Router Advertisement.*
    - "elementos_click": aquí podemos encontrar el código de los elementos de *Click Modular Router* implementados.
    - "proceso_auxiliar": donde se encuentra el código del proceso auxiliar de nivel de usuario y unos ejemplos de ficheros donde éste lee la versión, la potencia de transmisión, la prioridad de los paquetes, la posición inicial del nodo y el identificador de nivel de encaminamiento geográfico.

- Subdirectorio "Código_kernel": en este subdirectorio se encuentran los ficheros *if_arp.h* y *addrconf.c* del *kernel* de Linux que ha sido necesario modificar para la integración de IPv6 sobre el nivel de encaminamiento geográfico.

- Subdirectorio "Código_pruebas": aquí se encuentra el código de los programas generador_posiciones y analisis_resultados que se han utilizado en la evaluación de la implementación. También podemos encontrar los *scripts* de *Shell* de Linux utilizados en la máquina virtual estándar del vehículo A y en la máquina virtual especial para la repetición automática de la toma de muestras.

- Fichero PDF "Memoria": es la memoria del proyecto en formato PDF.

Apéndice F

Bibliografía

[1] “IEEE 802.11p”, http://en.wikipedia.org/wiki/IEEE_802.11p

[2] “Red Ad hoc”, http://es.wikipedia.org/wiki/Red_Ad_hoc

[3] Car-to-Car Communication Consortium, “C2C-CC Manifesto”, Versión 1.1, Agosto 2007, disponible en http://www.car-to-car.org/fileadmin/dokumente/pdf/C2C-CC manifesto 2007 09 24 v1.1.pdf.

[4] “Mission & Objectives”, http://www.car-to-car.org/

[5] C. Maihüfer, “A survey on geocast routing protocols”, IEEE Communications Surveys and Tutorials, 2nd quarter issue, vol. 6, no. 2, 2004.

[6] M.G.de la Fuente, H.Ladiod. “A Performance Comparison of Position-Based Routing Approaches for Mobile Ad Hoc Networks” en Vehicular Technology Conference, 2007. VTC-2007 Fall. 2007 IEEE 66th,Octubre 2007,pp.1-5.

[7] S. Basagni, I. Chlamtac, V. Syrotiuk, and B. Woodward, “A Distance Routing Effect Algorithm for Mobility (DREAM)”, en ACM/IEEE MOBICOM, 1998.

[8] Y.-B. Ko and N. H. Vaidya, “Geocasting in Mobile Ad Hoc Networks: Location-Based Multicast Algorithms”, Proc. 2nd Wksp. Mobile Comp. Sys. and Applications (WMCSA '99), New Orleans, USA, Febrero 1999, pp. 101–10.

[9] P. Yao, E. Krohne, and T. Camp, “Performance comparison of geocast routing protocols for a MANET,” en Proceedings of the 13th IEEE International Conference on Computer Communications and Networks (IC3N), Chicago, IL USA, Octubre 2004, pp. 213–220.

[10] A. Capone, M. Garcia de la Fuente, L. Pizziniaco, I. Filippini, “SIFT: An efficient method for Trajectory Based Forwarding”, en IEEE International Symposium on Wireless Communications Systems (ISWCS’05), Septiembre 2005.

[11] W.-H. Liao et al., "GeoGRID: A Geocasting Protocol for Mobile Ad Hoc Networks Based on GRID," J. Internet Tech., vol. 1, no. 2, Diciembre 2000, pp. 23–32.

[12] J. C. Navas and T. Imielinski, “Geographic Addressing and Routing”, Proc. 3rd ACM/IEEE Int'l. Conf. Mobile Comp. and Net. (MobiCom '97), Budapest, Septiembre 1997.

[13] B. Karp and H. T. Kung. “GPSR: Greedy perimeter stateless routing for wireless networks”. En Proceedings of the Sixth Annual ACM/IEEE International Conference on Mobile Computing and Networking (Mobicom 2000), Agosto 2000.

[14] “GeoNet Project”, http://www.geonet-project.eu/

[15] GeoNet, “GeoNet D2.1 Specifications- 1st realease”,Versión 1.2,Marzo 2009, disponible en http://www.geonet-project.eu/?download=GeoNet-D2.1-specification.pdf.

[16] Roberto Baldessari, Carlos J. Bernardos, María Calderón, “GeoSAC- Scalable Address Autoconfiguration for VANET Using Geographic Networking Concepts”, PIMRC 2008 Special Session 5a: Intelligent Transportation and Traffic Telematics – Challenges, Applications, Standariztion (invited paper), Cannes, Francia, 17 Septiembre 2008.

[17] Young-Jin Kim, Ramesh Govindan, Brad Karp and Scott Shenker. “On the Pitfalls of Geographic Face Routing”. En The Proceedings of the Proceedings of the Third ACM/SIGMOBILE International Workshop on Foundations of Mobile Computing (DIAL-M-POMC 2005), Septiembre 2005.

[18] V. Devarapalli, R. Wakikawa, A. Petrescu, and P. Thubert, “Network Mobility (NEMO) Basic Support Protocol,” RFC 3963 (Proposed Standard), Enero 2005.

[19] S. Thomson, T. Narten, and T. Jinmei, “IPv6 Stateless Address Autoconfiguration,” RFC 4862 (Draft Standard), Septiembre 2007.

[20] Donzé, François, “IPv6 Autoconfiguration”, The Internet Protocol Journal, vol. 7, no. 2, Cisco Systems. 21 Abril 2008, disponible en http://www.cisco.com/web/about/ac123/ac147/archived_issues/ipj_7-2/ipv6_autoconfig.html

[21] Iljitsch van Beijnum, “IPv6 Internals”, The Internet Protocol Journal, vol. 9, no. 2, Cisco Systems, disponible en http://www.cisco.com/web/about/ac123/ac147/archived_issues/ipj_9-3/ipv6_internals.html

[22] T. Narten, E. Nordmark, W. Simpson and H. Soliman, “Neighbor Discovery for IP version 6 (IPv6)”, RFC 4861 (Standards Track), Septiembre 2007.

[23] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek, “The click modular router”, ACM Transactions on Computer Systems., vol. 18, no. 3, pp. 263–297, 2000.

[24] Eddie Kohler. “The Click modular router”. PhD thesis, Massachusetts Institute of Technology, Noviembre 2000.

[25] “The Click Modular Router Project”, http://read.cs.ucla.edu/click/

[26] Joachim Klein, “Implementation of an ad-hoc routing module for an experimental network”, Master thesis, Universität Stuttgart and the Universidat Politècnica de Catalunya, Abril 2005.

[27] “Linux IPv6 Router Advertisement Daemon (radvd)”, http://www.litech.org/radvd/

[28] “Kernel Korner -Why and How to Use Netlink Socket”, http://www.linuxjournal.com/article/7356

[29] “VMWare Server,free virtualization download for virtual server consolidation”, http://www.vmware.com/products/server/

[30] “VNUML-WIKI”, http://www.dit.upm.es/vnumlwiki/index.php/Main_Page

[31] “Precision Time Protocol”, http://en.wikipedia.org/wiki/Precision_Time_Protocol

[32] “PTPd-Precision Time Protocol daemon”, http://ptpd.sourceforge.net/

[33] “NEC Global”, http://www.nec.com/

[34] Panoskrt, “Clone virtual machine onVMWare Server 2.0”, Enero 2009, http://panoskrt.wordpress.com/2009/01/20/clone-virtual-machine-on-vmware-server-20/

[35] “Ubuntu-es. Portal hispano de Ubuntu”, http://www.ubuntu-es.org/

[36] Caribdis, “Como compilar el kernel de Ubuntu”, http://www.ubuntu-es.org/?q=node/431

[37] “The Linux kernel archives”, http://kernel.org/

[38] Malco2001, “HOWTO: NFS Server/Client”, Septiembre 2006, http://ubuntuforums.org/showthread.php?t=249889